Source Inspector - Uma Ferramenta para

Extração de Regras de Negócio em Sistemas Legados

José Roberto Brasileiro de Siqueira

Instituto de Matemática - IM

Núcleo de Computação Eletrônica - NCE

Universidade Federal do Rio de Janeiro – UFRJ

Mestre em Informática

Prof. Eber Assis Schmitz, Ph.D.
NCE/UFRJ

Rio de Janeiro, RJ – Brasil

Novembro - 2002

Source Inspector – Uma ferramenta para

Extração de Regras de Negócio em Sistemas Legados

José Roberto Brasileiro de Siqueira

Dissertação submetida ao corpo docente do Instituto de Matemática / Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro – UFRJ - como parte dos requisitos necessários à obtenção do grau de Mestre em Informática.

Aprovada por:

_____

Prof. Eber Assis Schmitz, Ph.D.
DCC-IM/NCE-UFRJ (Orientador)

_____

Prof. Alberto Sulaiman Sade Júnior, D.Sc.
Banco Central do Brasil

_____

Prof. Pedro Manoel da Silveira, Ph.D.
DCC-IM/NCE-UFRJ

_____

Profª . Maria Luiza Machado Campos, D.Sc.
DCC-IM/NCE-UFRJ
(Suplente)

Rio de Janeiro, RJ – Brasil

Novembro - 2002

Siqueira, José Roberto Brasileiro de.

Source Inspector – uma ferramenta para extração de regras de negócio em sistemas legados. Rio de Janeiro: UFRJ/IM/NCE, 2002

Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro / Instituto de Matemática / Núcleo de Computação Eletrônica, Rio de Janeiro, 2002. Orientador: Eber Assis Schmitz.

1-Regras de Negócio 2-Sistemas Legados 3-COBOL 4-

Para Cida, Ivan e Carolina.

Agradecimentos

Ao professor Eber Schmitz, pelo apoio e constante confiança na conclusão deste projeto.

Ao Banco Central do Brasil pela oportunidade para o desenvolvimento desta ferramenta.

Ao amigo e chefe da DICEL Ruben Almeida Galvão pela orientação técnica.

Ao ex-chefe do DEMAB, Eduardo Nakao, pela aprovação de minha licença e aos meus chefes atuais, Sergio Goldenstein e Ivan Luis, pelo apoio durante meu afastamento das atividades diárias do BC.

Ao colega de BC e amigo Dr. Alberto Sulaiman pelo apoio nos momentos difíceis da, muitas vezes penosa, travessia acadêmica.

Aos colegas do grupo do professor Eber sobre Regras de Negócio: Alexandre Dalleore e Sergio Corrêa pelas opiniões e boas idéias.

A Ana Gavinho, a mãe do Source Inspector, pelo apoio e dedicação na transmissão do conhecimento sobre a ferramenta.

Ao meu camarada Luis Campos da Natividade, cuja colaboração tornou possível a construção da nova versão do Source Inspector.

Aos colegas do laboratório do NCE, capazes de compartilhar horas de silêncio e ainda nos estimular com bom humor e confiança para alcançar o objetivo final.

Ao Sergio Guedes, pelo curso de linguagem C, sem o qual não conseguiria fazer qualquer implementação.

Aos meus sogros pelo ininterrupto suporte moral e material ao longo destes anos de Mestrado.

Às professoras Julia Strauch e Maria Luiza Campos pelo apoio e crítica construtiva ao texto.

A D. Deise e sua equipe de secretaria por desatarem os mais tenebrosos nós burocráticos.

Ao professor Sergio Schneider e sua esposa, minha amiga Rosa, pelo constante incentivo, pela revisão cuidadosa e pelo amparo intelectual.

Ao amigo Francisco Vidinha, que mesmo estando à frente de um projeto crítico, foi generoso em compartilhar comigo um pouco de seus conhecimentos sobre regras de negócio do Selic e sobre as melhores práticas de programação em um ambiente UNISYS.

RESUMO

SIQUEIRA, José Roberto Brasileiro de. **Source Inspector – uma ferramenta para extração de regras de negócio em sistemas legados;** Orientador: Eber Assis Schmitz. Rio de Janeiro : UFRJ/NCE; novembro/2002. Dissertação (Mestrado em Informática).

Regras de negócio e sistemas legados são objetos de pesquisa de muitos centros acadêmicos e comerciais ligados à Tecnologia da Informação. No caso das regras de negócio, trata-se de identificar e explicitar aspectos primordiais do funcionamento das organizações. Os sistemas legados, por sua vez, constituem uma parte significativa da explicitação do conhecimento, embutindo, mesmo que de forma difusa e desorganizada, as próprias regras de negócio. O Banco Central do Brasil possui um estratégico acervo legado em ambiente UNISYS e, embora muitos estudos e ferramentas tenham sido desenvolvidas para tratamento de sistemas legados, poucas delas cobrem completamente a extração de regras de negócio em ambientes UNISYS contemplando alguma proposta de classificação para melhor identificação das regras obtidas. O Source Inspector é uma ferramenta desenvolvida no Núcleo de Computação Eletrônica da Universidade do Rio de Janeiro (NCE/UFRJ) que apresenta uma proposta de modelo de classes a partir de sistemas legados em UNISYS que possuam bancos de dados DMSII e código fonte COBOL. Este trabalho consiste da especificação e implementação de um conjunto de heurísticas para recuperar regras de negócio de sistemas em produção com o objetivo de promover uma melhor compreensão do ambiente legado.

ABSTRACT

SIQUEIRA, José Roberto Brasileiro de. **Source Inspector – uma ferramenta para extração de regras de negócio em sistemas legados;** Orientador: Eber Assis Schmitz. Rio de Janeiro : UFRJ/NCE; julho/2002. Dissertação (Mestrado em Informática).

Business rules and legacy systems have been studying by many research centers linked to information technology. The business rules try to identify and become explicit the main aspects of the organization's life. On the other hand, legacy systems form a remarkable and explicit portion of the organizational knowledge. Most of the explicit business rules, although spread and disorganized, belong to the legacy code. Central Bank of Brazil has an strategic legacy asset in UNISYS environment. Although many works and tools have been released in the market, very few of them covers completely the business rule extraction issue in UNISYS environment along with some taxonomy proposition in order to better identify business rules. Source Inspector is a tool conceived in NCE/UFRJ to propose a class model from a UNISYS legacy system based upon DMSII databases definitions and COBOL source code. This dissertation proposition is the specification and implementation of a set of heuristics to recover business rules from real-life code in order to allow a better legacy environment understanding.

Lista de Siglas e Abreviaturas

ANDIMA	-	**A**ssociação **N**acional **D**os **I**nvestidores do **M**ercado **A**berto
BC	-	**B**anco **C**entral do Brasil
BD	-	Banco de Dados
CICS	-	**C**ustomer **I**nformation **C**ontrol **S**ystem
COBOL	-	**CO**mmon **B**usiness **O**riented **L**anguage
CODASYL	-	**C**onference **O**n **DA**ta **SY**stem **L**anguages
CPD	-	**C**entro de **P**rocessamento de **D**ados
DASDL	-	**D**ata **A**nd **S**tructure **D**efinition **L**anguage
DBA	-	**D**ata **B**ase **A**dministrator
DEMAB	-	**DE**partamento de Operações do **M**ercado **AB**erto
DMSII	-	**D**ata **M**anagement **S**ystem, versão **II**
HRN	-	**H**eurística de **R**egra de **N**egócio
ISO	-	**I**nternational **O**rganization for **S**tandardization
OO	-	**O**rientação a **O**bjeto
RHA	-	**R**egras **H**abilitadoras de **A**ção
RN	-	**R**egras de **N**egócio
SBES	-	**S**impósio **B**rasileiro de **E**ngenharia de **S**oftware
SELIC	-	**S**istema **E**special de **LI**quidação e **C**ustódia
SERPRO	-	**SER**viço Federal de **PRO**cessamento de Dados
SFN	-	**S**istema **F**inanceiro **N**acional
SGBD	-	**S**istema **G**erenciador de **B**anco de **D**ados
SQL	-	**S**tructured **Q**uery **L**anguage
TCP/IP	-	**T**ransfer **C**ontrol **P**rotocol / **I**nternet **P**rotocol
TI	-	**T**ecnologia da **I**nformação
UML	-	**U**nified **M**odelling **L**anguage
YACC	-	**Y**et **A**nother **C**ompiler-**C**ompiler

Índice de Figuras

Pág.

Figura 1.1 – Esquema simplificado da distribuição do conhecimento pela organização e da importância do código legado neste contexto. 2

Figura 2.1 – Ilustração sobre a compatibilidade entre a explicitação das RN no código legado e na documentação. 11

Figura 2.2 – Estrutura hierárquica de um programa Cobol.....13

Figura 2.3 – Extração de modelo de negócios a partir de sistemas legados 30

Figura 3.1 - Diagrama da estrutura do DMSII.....34

Figura 3.2 - Diagrama da estrutura de um programa Cobol.....40

Figura 3.3 – Hierarquia de regras de negócio na extração dos fontes Unisys.....42

Figura 3.4 – Diagrama de atividades descrevendo a fase 1.....43

Figura 3.5 – Diagrama de atividades descrevendo a fase 2.....44

Figura 3.6 – Diagrama de atividades descrevendo a fase 3.....45

Figura 3.7 – Diagrama de atividades descrevendo a fase 4.....46

Figura 4.1 - Estrutura interna do Source Inspector.....59

Figura 4.2 – Parametrização para carga inicial de um fonte no Source Inspector.....63

Figura 4.3 – Visualização do Source Inspector após a carga do DBLEINF.....66

Figura 4.4 – Visualização do Source Inspector após a carga do programa LEINF-OBJ-LI002000, pertencente ao sistema LEINF.....67

Figura 4.5 – Seqüência de seções e parágrafos.....74

Figura 4.6 – Árvore gerada pelo ponteiro *pt_chamada_sec_par*.....75

Figura 4.7 – Interface para a o procedimento de equivalência de termos.....82

Figura 4.8 – Profundidade de chamada para os parágrafos.....92

Figura 4.9 – Simbologia utilizada no Source Inspector para caracterizar as regras de negócio na interface.....96

Figura 4.10 – Termos de negócio extraídos do DBLEINF.....97

Figura 4.11 – Termos do programa LEINF-OBJ-002000.....98

Figura 5.1 – Interface do Source Inspector com a tela para edição de documentação das regras.....103

Figura 5.2 – Modelador de Empresas.com participação do Source Inspector.....108

Figura A.1 – Política econômica e atuação do SELIC na política monetária.....120

Figura A.2 – Sistema Financeiro Nacional e identificação das Autoridades Monetárias..122

Figura A.3 – Localização na estrutura do BC.....125

Figura C.1 – Modelo simplificado contendo as tabelas de Natureza, Termos de Negócio e Padrões.....135

Figura C.2 – Interface do Source Inspector para cadastramento de naturezas.....138

Figura C.3 – Cadastramento de naturezas para o banco DBLEINF.....138

Figura C.4 – Cadastramento de padrões para banco de dados.....140

Figura C.5 – Cadastramento de padrões para o programa LEINF-OBJ-LI002000.....143

Figura C.6 – Cadastramento de padrões de parágrafo para o programa LEINF-OBJ-LI002000.....145

Figura C.7 – Exemplo de um item de grupo em fonte Cobol.....148

Figura C.8 – Árvore montada para a estrutura FUNCIONARIO na primeira passada do ModTra.....149

Figura D.1 – Esquema simplificado das etapas do processo de apuração.....151

Índice de Quadros

Pág.

Quadro 2.1 – Framework de Zachman.....23

Quadro 2.2 – Principais produtos comerciais que tratam sobre regras de negócio.....25

Quadro 3.1 – Descrição dos principais elementos utilizados pelo DASDL.....35

Quadro 3.2 – Descrição dos principais elementos utilizados pelo Cobol.....38

Quadro 3.3 - HRN para identificação e extração de termos de negócio definidas a partir do banco de dados DMSII.....49

Quadro 3.4 - Descrição da HRN derivadas de bancos de dados.....50

Quadro 3.5 – HRN obtidas a partir do tratamento das declarações da DATA-BASE SECTION.....52

Quadro 3.6 – HRN obtidas através do tratamento dos programas Cobol.....54

Quadro 3.7 - HRN para estabelecer abrangência dos itens de dados identificados no código do programa.....56

Quadro 3.8 - HRN para caracterizar os arquivos e datasets do código legado.....57

Quadro 4.3 – Resultado da execução do Módulo Tradutor DASDL sobre os fontes BC...67

Quadro 4.4 – Resultado da execução do Módulo Tradutor Cobol sobre os fontes BC.....67

Quadro 4.5 – Resultado da execução do Módulo Tradutor DASDL sobre os fontes DATAPREV.....68

Quadro 4.6 – Resultado da execução do Módulo Tradutor DASDL sobre o fonte do sistema PCP.....69

Quadro 4.7 – Heurísticas por regra de negócio.....76

Quadro 4.8 – Termos de negócio extraídos de DBOFPUB.....78

Quadro 4.9 – Termos de negócio extraídos da WS SECTION do programa ProcessaApuracaoDireta, utilizando a heurística HRNT03.....79

Quadro 4.10 – Relação final dos termos de negócio após a aplicação dos padrões.....81

Quadro 4.11 – Fatos extraídos pelo Source Inspector.....83

Quadro 4.12 – Restrições extraídas pelo SourceInspector.....84

Quadro 4.13 – Totalização das regras extraídas.....92

Quadro 5.12 – Totalização por heurística.....94

Quadro C.1 – Descrição da tabela de Termos de Negócio.....136

Quadro C.2 – Descrição da tabela de Naturezas de atributo.....136

Quadro C.3 – Exemplos de Natureza.....137

Quadro C.4 – Descrição da tabela de Padronização.....139

Quadro C.5 – Padrões cadastrados para um determinado banco de dados.....139

Quadro C.6 – Resultado da padronização sobre a extração de termos de negócio em bancos de dados.....140

Quadro C.7 – Descrição da tabela de Padronização para fontes Cobol.....141

Quadro C.8 – Padrões cadastrados para um determinado sistema.....142

Quadro C.9 – Resultado da padronização sobre a extração de termos de negócio em programas.....143

Quadro C.10 – Padronizações para os parágrafos.....144

Quadro C.11 – Possíveis tipos de cadastramento.....144

Quadro C.12 – Resultado da padronização sobre a extração de nomes de parágrafos e seções.....145

Quadro C.13 – Exemplo de arquivo intermediário a partir de um fonte DASDL.....146

Quadro C.14 – Detalhamento do tipo de registro 2 (CAMPO).....147

Quadro C.15 – Tipo de registro C2 após execução do ModTra.....148

Quadro D.1 – Resultado da apuração preliminar ordenando as propostas por PU.....153

Quadro D.2 – Grupamento por faixa de PU.....154

Quadro D.3 – Resultado final da apuração direta por instituição.....155

SUMÁRIO

Pág.

1 INTRODUÇÃO 1
1.1 Apresentação 1
1.2 Motivações 2
1.3 Objetivo 4
1.4 Delimitação 4
1.5 Trabalhos relacionados 5
1.6 Contribuições 5
1.7 Plano da Tese 6
2 SISTEMAS LEGADOS E REGRAS DE NEGÓCIO 8
2.1 Sistemas Legados 8
2.1.1 Conceituação 8
2.1.2 COBOL: origem e características 11
2.1.3 O que fazer com os sistemas legados ? 12
2.2 Regras de Negócio 14
2.2.1 Regras de Produção 14
2.2.2 Definição 15
2.2.3 Histórico 17
2.2.4 Classificação de RN 19
2.2.5 Produtos Comerciais 20
2.2.6 Identificação e extração de regras de negócio 22
2.3 Trabalhos correlatos 23
3 HEURÍSTICAS E O PROCESSO PARA IDENTIFICAÇÃO DE REGRAS DE NEGÓCIO 28
3.1 Um metamodelo DMSII/COBOL 29
3.1.1 Um metamodelo para os bancos de dados DMSII 29
3.1.2 Um metamodelo dos programas COBOL 31
3.2 Heurística e Tipos de RN identificadas 34
3.3 O processo de Identificação de Termos de Negócio 37
3.3.1 Fase 1 – Preparação dos fontes 42

3.3.2 42
3.3.3 Fase 2 – Cadastramento de parâmetros....42
3.3.4 Fase 3 – Extração das regras de BD 44
3.3.5 Fase 4 – Extração das regras de programa.....47
3.4 Source Inspector 54
3.4.1 O Módulo Tradutor (ModTra) 55
3.4.2 O Módulo Extrator (ModEx) 59
4 RESULTADOS DO ESTUDO DE CASO COM O SOURCE INSPECTOR....61
4.1 Validação da ferramenta 61
4.1.1 Obtenção de fontes e bancos de dados....61
4.1.2 Implementação das modificações no Módulo Tradutor....67
4.1.3 Implementação das heurísticas no Módulo Extrator....75
4.1.4 Demonstração do funcionamento do Source Inspector 76
4.2 Extração das Regras de Negócio 82
4.2.1 Extraindo Fatos 82
4.2.2 Extraindo Restrições 83
4.2.3 Extraindo Regras das Sentenças 84
4.3 Análise dos Resultados 94
4.3.1 Regras de negócio extraídas....94
4.3.2 Heurísticas utilizadas 96
4.3.3 Outros Benefícios do Source Inspector 98
5 CONCLUSÃO 103
5.1 Contribuições 103
5.2 Trabalhos Futuros 107
6 REFERÊNCIAS BIBLIOGRÁFICAS....113
ANEXO A - O SELIC NO SISTEMA FINANCEIRO NACIONAL 121
A.1 - Ambientação....121
A.2 - Instrumentos de Política Monetária....122
A.3 – Sistema Financeiro Nacional (SFN) 124
A.4 – O Mercado Aberto 128
A.5 – Um Sistema de Registro 129

ANEXO B – PADRÃO SELIC DE PROGRAMAÇÃO.....131
B.1 - Estrutura geral do programa COBOL.....131
B.2 - Implementação das estruturas básicas.....132
B.3 - Regras gerais de programação.....135
B.4 Variáveis.....137
ANEXO C – ESQUEMA DAS PADRONIZAÇÕES PARA IDENTIFICAÇÃO DE TERMOS DE NEGÓCIO.....139
C.1 - Termo de Negócio.....140
C.2 - Natureza.....140
C.3 - Padrões de BD.....145
C.4 - Padrões para programas.....147
C.5 - Padrões de parágrafos.....150
ANEXO D - APURAÇÃO DIRETA NO SISTEMA OFPUB.....156
D.1 - Relatórios do Sistema Oferta Pública Formal Eletrônica (OFPUB).....156
D.2 - Comunicado/Portaria.....156
D.3 - Apuração.....157
D.4 - Exemplo Prático de uma Apuração Preliminar:.....159
D.5 Comentários sobre a apuração.....161
ANEXO E - DESCRIÇÃO DO BANCO DE DADOS DBOFPUB NO FORMATO DASDL 163
ANEXO F - DESCRIÇÃO DO BANCO DE DADOS DBOFPUB NO FORMATO DASDL 166

1 INTRODUÇÃO

"Aqui na empresa, em muitas áreas,
o conhecimento sobre o negócio
perdeu-se e os sistemas
estão tocando o negócio."
Constatação de um gerente resignado

"Não sei se o sistema vai aceitar isto."
Resposta padrão para solicitações fora do padrão

1.1 Apresentação

Este trabalho mostra o resultado de um estudo sobre heurísticas de extração de regras de negócio em sistemas legados, os quais representam uma das formas de explicitação de conhecimento ao relacionar linhas de código a alguma regra de negócio mapeada, seja para auxiliar a tomada de uma decisão ou apoiar uma atividade de manutenção ou desenvolvimento de aplicativos. A Figura 1-1 ilustra algumas formas de conhecimento, dentre as quais o código legado.

Um grande desafio para os gestores de TI e particularmente para os responsáveis pelos sistemas de informação de uma organização vem sendo a recuperação deste conhecimento explícito em programas antigos, escassamente

documentados e significativamente alterados desde sua primeira versão [GAVI00, HSMA00, SIQU00, SIQU01, SIQU02].

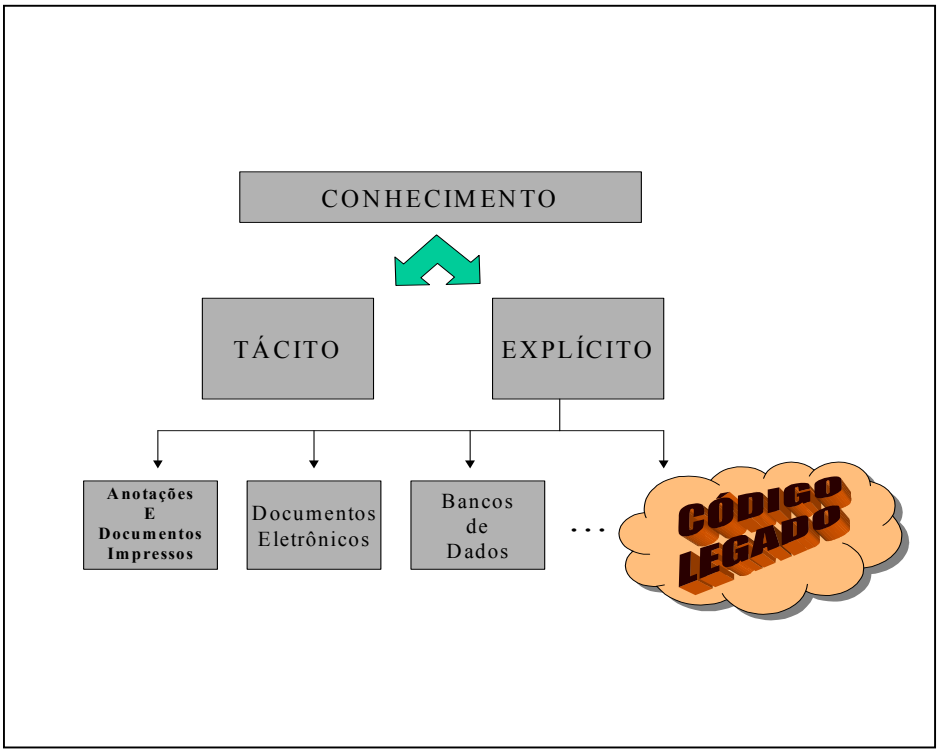

Figura 1-1 - Esquema simplificado da distribuição do conhecimento explícito pela organização e da importância do código legado neste contexto. (Fonte: [SIQU01])

1.2 Motivações

Introduzir e disseminar o conceito de regra de negócio no BC é visto como ponto estratégico para a organização. Regras de negócio permitem uma rápida mudança na logística do núcleo do negócio (*core logistics*), especialmente se houver integração com os sistemas computacionais que as suportam. Num ambiente volátil com o do mercado financeiro, trabalhar com as regras de negócio constitui uma oportunidade para repensar o contexto do mercado, avaliando riscos, táticas e políticas [LAM00, ZACH00].

No ambiente do Banco Central do Brasil encontramos diversos sistemas que se enquadram no perfil de software legado, o que vem motivando seus

gerentes de TI a encontrar soluções para compreensão deste código tendo em vista futuras manutenções ou migrações.

O **SELIC** rege o funcionamento do mercado aberto de títulos públicos federais, sendo apoiado por um sistema de informação homônimo implantado no final da década de 70[1]. Tanto o **SELIC** como seus subsistemas rodam em ambiente UNISYS, com fontes COBOL acessando estruturas de banco de dados DMSII. Recentemente, a implantação do novo Sistema de Pagamentos Brasileiro demandou inúmeras alterações no **SELIC** em caráter de urgência. O **SELIC** é um software extremamente gasto. Em seus mais de vinte anos de vida, este sistema testemunhou o período mais instável da economia brasileira, tendo que se adaptar a diferentes planos econômicos, padrões de moeda e correções diversas. Estas alterações em um sistema desenvolvido com pouco método e sem um processo de qualidade definido tornaram a implementação de qualquer solução nova ou requisição dos clientes algo extremamente trabalhoso [PRESS97, BRAN97].

No **SELIC**, as regras de negócio mais confiáveis estão escritas em código COBOL. Apesar do bom desempenho do ambiente UNISYS, sua arquitetura proprietária exige uma série de recursos adicionais para que aplicações com tecnologia *Web*, por exemplo, possam ser implementadas. Além disso, a própria estrutura hierárquica do DMSII também possui limitações que dificultam a manipulação de seus dados, exigindo recursos e configurações específicas dos softwares de middleware, tornando esta atividade muito custosa [UNIS97b].

A documentação do **SELIC** está, em sua maior parte, incorporada ao seu código fonte. Muita informação existe apenas na cabeça dos mantenedores do sistema, já que muitos dos analistas que o conceberam não se encontram mais nos quadros do BC. Portanto, qualquer entendimento ou esclarecimento sobre o funcionamento do sistema não é trivial.

---

[1] Maiores detalhes sobre o **SELIC** no **Anexo A**.

A utilização de ferramentas para administração do código legado em ambiente UNISYS de modo geral e em bancos de dados DMSII de modo particular é limitada. O mercado para estas ferramentas é o do ambiente IBM que, por possuir uma base instalada bem maior apresenta melhores perspectivas de negócio para as empresas de software construirem suas soluções de extração. Portanto, boa parte das ferramentas existentes para tratamento de linguagens legadas funcionam em ambiente UNISYS a partir de adaptações, já que não foram concebidas com este objetivo. Os complexos sistemas existentes não apenas no Banco Central, mas em outras organizações nacionais de relevância como na Dataprev, Iplan-Rio, Andima, Cetip e Datamec, justificam plenamente o desenvolvimento de uma solução customizada para a plataforma UNISYS.

1.3 Objetivo

O objetivo deste trabalho é extrair, classificar e armazenar regras de negócio existentes em sistemas legados desenvolvidos em COBOL/DSMSII no ambiente UNISYS, facilitando assim a compreensão do patrimônio legado de software.

Descreveremos um processo para extração das regras de negócio utilizando heurísticas específicas para este fim. Os termos de negócio e as demais regras obtidas formam um repositório através do qual a equipe mantenedora do software legado e os especialistas de domínio podem compreender melhor seu acervo explícito em linguagem de programação.

1.4 Delimitação

Esta dissertação limita sua abordagem sobre o ambiente UNISYS em sistemas desenvolvidos com código COBOL e utilizando banco de dados DMSII. Não pretendemos especificar um ambiente de manutenção de programas nem propor formas de representação de regras.

A escolha de um ambiente específico surgiu com o objetivo de estabelecermos um foco que, se por um lado restringe a abrangência da solução, por outro a torna mais customizável e útil para a instalação.

1.5 Trabalhos relacionados

A primeira dissertação correlata a este trabalho data de 1997, onde [VILA97] define um processo de reengenharia que recupera um Modelo de Objetos a partir de um sistema COBOL construído segundo o paradigma estruturado. Analisando os principais programas de um sistema, a partir do código fonte são recuperados possíveis componentes OO. Em [GAVI00] encontramos uma continuação deste trabalho onde heurísticas são definidas para tratar a linguagem de especificação de banco de dados DMSII e, a partir daí, propor classes para transformação, mas sem abordar especificamente regras de negócio.

Em [SULA02] encontramos um trabalho aplicado em ambiente UNISYS e voltado para identificação de seqüências de regras endógenas nas transações com bancos de dados a partir dos logs gerados pelo DMSII. Em [CORR02] encontramos uma proposta para armazenamento de regras declarativas, permitindo assim outras formas de representação de regras, como por exemplo lógica de primeira ordem, que não fossem simplesmente as linhas de código.

1.6 Contribuições

Este trabalho contribui com a definição de heurísticas para identificação e classificação de regras de negócio. Estas heurísticas estão inseridas em um processo interativo e semi-automático para extração de regras de negócio apoiados pela ferramenta **Source Inspector**, um produto mais amadurecido e capaz de interpretar bem complexos códigos no ambiente UNISYS.

O trabalho também contribui para uma melhor compreensão do código legado, seja para documentação, entendimento do negócio, treinamento de novos colaboradores ou futura migração para outros ambientes.

Outra contribuição está na apresentação de uma alternativa simples para classificação de termos e regras de negócio de forma a facilitar a introdução da abordagem de regras de negócio em uma organização;

1.7 Plano da Tese

Na seção 2 será feita uma breve revisão de literatura abordando os temas sistemas legados e regras de negócio, acrescentando os trabalhos mais relevantes no contexto desta dissertação.

Na seção 3 descreveremos o processo para identificação das regras de negócio, apresentando o ambiente UNISYS, as etapas estabelecidas para a extração, as heurísticas utilizadas e o **Source Inspector,** o software desenvolvido para a solução do problema de extração. Na seção 4 apresentaremos um estudo de caso utilizando o processo e a ferramenta descrita. Na seção 5 encontra-se a conclusão com as contribuições obtidas e os trabalhos futuros que podem ser desenvolvidos.

Ao final deste trabalho foram incluídos seis anexos. O primeiro deles traz uma explicação mais detalhada sobre o SFN, sua estrutura, a importância do BC e o papel do **SELIC**. Os anexos seguintes tratam de padronização: o anexo B apresenta o padrão de programação utilizado pelos analistas mantenedores dos fontes do **SELIC**, enquanto o anexo C descreve os padrões que podem ser parametrizados pelos especialistas com o objetivo de otimizar o processo de extração de regras de negócio do **Source Inspector**. O anexo D descreve a rotina utilizada no experimento, enquanto o anexo E traz a descrição em código fonte do banco de dados utilizado. O anexo F traz maiores detalhes sobre a linguagem COBOL

2 SISTEMAS LEGADOS E REGRAS DE NEGÓCIO

With all the server advances lately,
pretty soon they're going to
invent the mainframe.
Jim Moore

2.1 Sistemas Legados

2.1.1 Conceituação

Sistemas legados são, por definição, aplicações de software existentes nas organizações e vitais para o funcionamento do negócio [SAYL96]. Costumam conter programas grandes, complexos e mal documentados [CHAP96]. Do ponto de vista do negócio em si, são considerados sistemas que apresentam muitas dificuldades para serem modificados ou incrementados para atenderem às constantes requisições e novas solicitações do negócio [BROD95 *apud* KRUC01]. As regras de negócio embutidas nestes sistemas muitas vezes não são reconhecidas nem pelos analistas mantenedores do código devido às inúmeras alterações que o software sofreu ao longo de seu ciclo de vida [GAVI00].

Quando as empresas, a partir dos anos 60, iniciaram a automação de seus sistemas, o objetivo era a redução de custos e de erros provocados por procedimentos mecânicos e repetitivos, executados manualmente. Desde então testemunhamos concepção e implantação de soluções de Informática, formando um extenso e complexo legado de software quase sempre crítico para a organização. [HAY99, GAVI00, SAYL96].

Nos anos 70, conceitos como modelagem de dados, abordagens *top-down,* programação estruturada e fluxo de dados ainda eram experimentais. Os programadores simplesmente traduziam especificações de negócio diretamente para código fonte. Esta prática, com o passar dos anos, foi fazendo com que as sucessivas manutenções revestissem o código gerado com complexas estruturas de programação, criando uma camuflagem cada vez mais impenetrável da lógica do negócio, que representa, em última análise, o coração operacional de uma empresa. Com isso, cada pequena alteração contribuía para aumentar o grau de deterioração dos sistemas, num verdadeiro efeito entropia [SEEC00, PRES97].

A conseqüência, como a Figura 2.1 tenta demonstrar, foi um gradual distanciamento entre a documentação das RN e sua transcrição para o código dos programas. Ou seja, ao longo do ciclo de vida do *software*, vão sendo implementadas uma série de alterações devido às mudanças que o próprio ambiente empresarial vai sofrendo, afastando cada vez mais o código legado dos requisitos originais do negócio expressos na documentação inicial do sistema. Conseqüentemente, o *único* lugar onde as RN existem explicitamente, muito provavelmente passa a ser apenas o código legado [HUAN98, ROSS00].

As RN transcritas diretamente para o código geram ainda uma forte dependência da memória viva dos desenvolvedores tornando difícil a existência, dentro de uma mesma organização, de uma pessoa ou grupo que compreenda bem tanto o software quanto a documentação existente [HUAN98].

Com as RN fragmentadas no sistema legado, os responsáveis pela manutenção de um software confiam mais no código programado do que nas regras documentadas, mesmo não sendo capazes de extraí-las dos fontes de forma confiável [HUAN98].

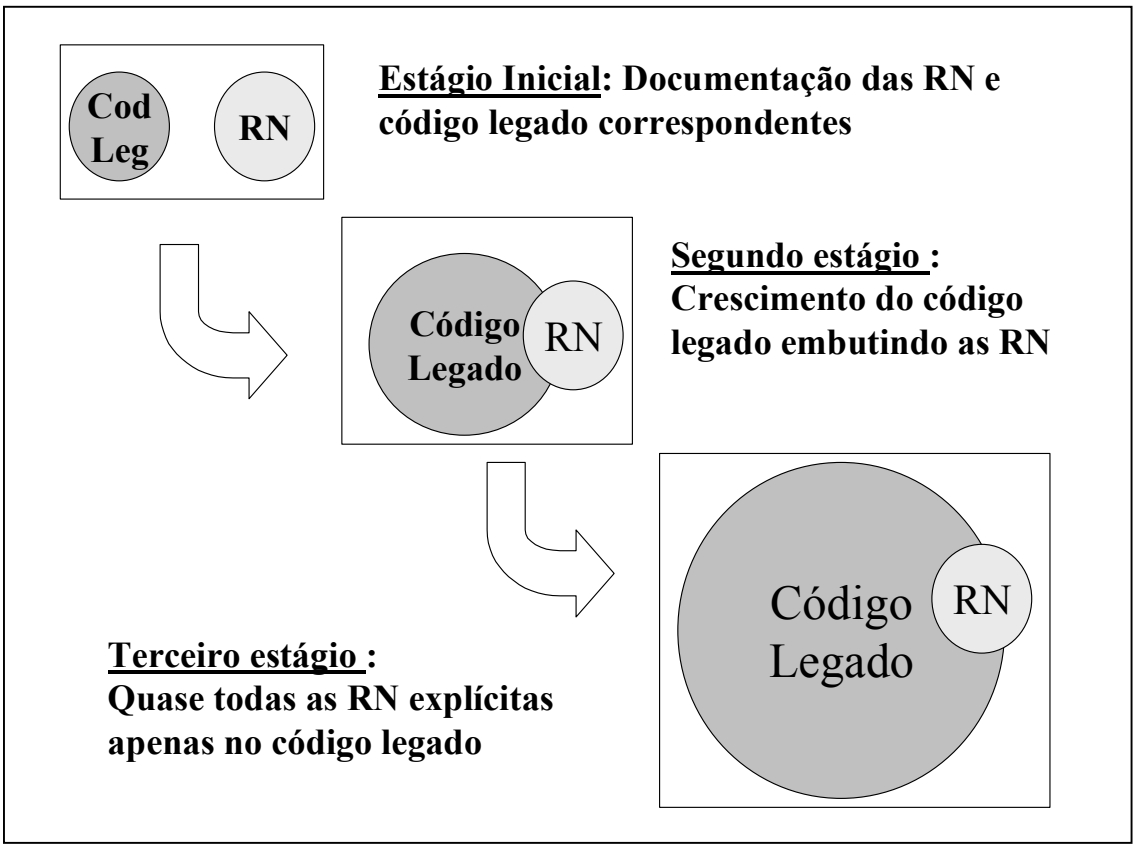

Figura 2-1 - Ilustração sobre a compatibilidade entre a explicitação das RN no código legado e na documentação. (Fonte: [SIQU01])

O esforço de manutenção do código legado se transforma em uma atividade custosa e arriscada, seja para corrigir erros (***manutenção corretiva)*****,** incorporar novas funções (***manutenção evolutiva ou perfectiva***), adaptar a aplicação para novas tecnologias e equipamentos (***manutenção adaptativa***) ou melhorar a estrutura interna do código gerado tendo em vista evitar problemas maiores no futuro (***manutenção preventiva***) [PRES97, BENN96].

2.1.2 COBOL: origem e características

Um ponto que merece particular atenção é a predominância do COBOL como linguagem de programação de aplicativos comerciais. Nos primórdios da computação comercial, não existiam muitas linguagens de programação que pudessem atender às restrições tecnológicas vigentes. PL1, Fortran, ALGOL e até mesmo Assembler, esta por requisitos estritamente de performance, foram bastante utilizadas, mas nenhuma delas superou o COBOL em termos de linhas de código implementadas em produção [SIQU00].

A CODASYL surgiu no final dos anos 50 através de uma iniciativa de vários fornecedores e usuários com o objetivo de criar uma linguagem de programação padronizada e não-proprietária, transformando-se numa das primeiras tentativas da indústria de Informática para estabelecer alguma padronização. Surgiu assim o COBOL que, ao longo da década de 60 sofreu diversas melhorias até que em 1968 nasceu a primeira versão padronizada pela ISO: o COBOL68. Desde então, sucessivos progressos foram feitos originando os consagrados padrões COBOL74 e COBOL85.

O COBOL visava atender a três premissas básicas: ser baseada na língua inglesa, a fim de que pudesse ser facilmente lida, ser direcionada para programação de procedimentos administrativos e ser implementável em qualquer plataforma e de fácil manutenção[2] [SELL97].

[2] Maiores detalhes sobre COBOL no **Anexo F**.

2.1.3 O que fazer com os sistemas legados ?

O competitivo mundo dos negócios vem pressionando as organizações a desenvolver aplicativos e implementar funções para atender a novos requisitos de negócio. Estas solicitações entretanto, não significam necessariamente a chegada de mais recursos para a área de sistemas. Desta forma, os gerentes de Informática são obrigados a dividir suas atenções entre os novos produtos, que vão trazer mais prestígio para a área ao apoiar novas oportunidades de negócio; e os sistemas legados, que representam a sustentação da empresa e, de forma alguma, podem deixar de funcionar [PAUL99].

Durante muito tempo na história do desenvolvimento de sistemas, as manutenções evolutivas representaram a melhor alternativa para implementar novas funções. Uma equipe experiente, de posse das especificações necessárias acrescentava e removia código com a maior tranqüilidade. Além da implementação mais rápida, esta opção parecia ser muito mais segura que enveredar pelos desconhecidos caminhos de um novo projeto, especialmente quando isto implicava migração ou mudança de ambiente operacional. Infelizmente, a maior parte destas "evoluções" foi implementada com pouco ou nenhum método, tornando as futuras modificações cada vez mais penosas [BRAN97, PRES97].

Esta situação foi chegando a um ponto crítico onde a decisão de se construir um sistema inteiramente novo, ignorando quase completamente o sistema existente em produção, passou a ser considerada uma alternativa viável. Intuitivamente, "abandonar o velho e adotar o novo" parecia um raciocínio mais razoável. Esta tendência chegou a ser amplamente divulgada na literatura técnica e apoiada por muitas consultorias externas [HAMM01, HAMM02].

Entretanto, uma questão passou a incomodar os analistas mais experientes. Desprezar completamente o sistema atual não estaria sendo um desperdício? Muitas regras de negócio embutidas no código fonte não poderiam

representar rica fonte de informação e requisitos para a especificação do novo sistema? No código legado não estariam registradas as prioridades, desejos e comportamentos dos usuários? A nova equipe estaria realmente livre de incorrer nos mesmos erros e cair nas mesmas armadilhas da equipe responsável pelo antigo projeto [KUCH01, SEEC00] ?

Estas questões permaneceram em aberto. Manter o ambiente legado e cuidar do patrimônio de software, portanto, passou a ser visto desde então como uma atividade crítica para qualquer empresa, pois processos de negócio estratégicos, representando significativas vantagens competitivas, estão embutidos nos códigos das aplicações legadas. Além disso, o valor percebido deste patrimônio nas organizações aumentou muito, especialmente após a virada do ano 2000 [HEIM00, SEEC00].

Portanto, para os gestores de Informática que lidam com excesso de demanda e recursos escassos, a opção de migrar os sistemas legados para outra plataforma, iniciando seu desenvolvimento do ponto inicial e desprezando os sistemas legados, não se torna viável, mesmo se considerarmos o risco de um sistema legado conter uma série de vícios antigos, dificultando o exame e a consideração de novas abordagens. Na verdade, a produção de uma nova aplicação do ponto inicial pode ser encarada como um problema financeiro de investimento e retorno cujo risco associado não pode ser assumido [KUCH01, SIQU00].

Uma das alternativas mais utilizadas para solucionar esta questão é a renovação de sistemas ligados, o que segundo [DEUR99], consiste em "aplicar uma tecnologia do *futuro* com o objetivo de trazer os sistemas do *passado* para uma utilização no *presente*", adaptando-os a uma nova conjuntura. Segundo [HEIM00] podemos, neste contexto, definir três "Ms" para a renovação de aplicações legadas: modernização, modularização e migração.

A **Modernização** implementa mudanças mínimas, também denominadas cosméticas, sem a necessidade de reescrever o código da aplicação, mas suficientes para que as aplicações desenvolvidas com tecnologias de décadas atrás sejam beneficiadas pelos avanços tecnológicos e possam atender a novos requisitos. Esta abordagem gera uma segurança muito maior já que sem uma mudança no software, as funcionalidades são preservadas e não é preciso fazer uma análise mais profunda das estruturas e do código legado. A modernização apresenta baixo custo (não há necessidade de novo hardware nem novos equipamentos) e requer menor investimento em futuros treinamentos junto aos usuários (interface graficamente mais adequada e intuitiva) [KUCH01].

A **Modularização** refere-se à identificação, compreensão e extração de metadados e regras de negócio a partir dos dados e códigos dos sistemas legados, o que é a chave para a alavancagem do patrimônio legado. O resultado final desta atividade tremendamente trabalhosa é a construção de objetos (componentes) bem definidos e capazes de ser reutilizados. A maior dificuldade é a necessidade de uma análise mais profunda, demorada e dispendiosa para compreensão do código legado.

Finalmente, a **Migração** é o uso das informações obtidas com a modularização para modificar sistemas existentes ou construir novos aplicativos, adaptando-os a novas plataformas. Via de regra a migração torna-se necessária quando a vida útil do sistema ultrapassou a do hardware, sistema operacional ou middleware sobre o qual foi implementado [KUCH01].

2.2 Regras de Negócio

2.2.1 Regras de Produção

Qualquer programa encerra, em seu código e em suas estruturas de dados, algum conhecimento sobre o domínio do problema que soluciona. Este conhecimento se localiza basicamente nos procedimentos de decisão, que

determinam a seqüência de execução dos algoritmos invocados e se manifestam nas diferentes saídas obtidas para cada entrada específica. Boa parte do restante do conhecimento sobre o domínio de um problema entretanto, está com os especialistas que, para transmiti-lo, utilizam-se de algum recurso de representação do conhecimento que pode apresentar menor ou maior rigor formal (linguagem corrente ou estruturada, diagramas UML, redes semânticas, lógica matemática) [HAY99, HEIM00, SELL97, VERH00].

Regras de produção (através de pares "condição-ação") se constituem na maneira mais utilizada de representar conhecimento. Filosoficamente, optar pela sua utilização para representar o conhecimento de um especialista significa acreditar que, ao menos em parte, as pessoas raciocinam em função de um conjunto de regras dedutivas, as quais proporcionam a habilidade de derivar nova informação a partir da informação preexistente [SULA00].

Uma ação consiste de um conjunto de declarações imperativas da linguagem, podendo ser sucessivamente decomposta até que se chegue a uma regra elementar. Estes níveis podem ser deduzidos da própria estratégia utilizada no desenvolvimento de programas estruturados, partindo-se do genérico para o específico. Este tipo de modelo também é utilizado na prática de solução de problemas.

2.2.2 Definição

A definição de regra de negócio pressupõe uma conceituação do termo *regra*. Na língua portuguesa, podemos encontrar as seguintes definições mais compatíveis com nosso assunto [FERR86]:

1. Regra é aquilo que regula, dirige, rege ou governa;

2. Regra é fórmula que indica ou prescreve o modo correto de falar, de pensar, raciocinar, agir, num caso determinado;
3. Regra é aquilo que está determinado pela razão, pela lei ou pelo costume; preceito, princípio, lei, norma.

As RN podem ser definidas sob duas perspectivas: a do negócio e a dos sistemas de informação. Do ponto de vista do negócio, trata-se de uma abordagem que envolve fortemente o fator humano existente no sistema, sendo definida como "uma diretiva com o objetivo de influenciar ou guiar o comportamento do negócio, apoiando as políticas formuladas em função das oportunidades e ameaças do ambiente no qual a organização está inserida" [BUSI99]. Em última análise, RN são derivadas das políticas de negócio (missão, estratégia) formuladas para refletir influências (tendências), aproveitar oportunidades e considerar riscos e ameaças [ROSS00, CHAP02].

Na perspectiva dos sistemas de informação, RN são "declarações que definem ou qualificam algum aspecto da estrutura do negócio controlando ou influenciando algum comportamento, representando o conhecimento dos especialistas do negócio" [GUID97, BUSI99, ROSS00]. Por estrutura do negócio podemos considerar o modelo estático do negócio com seus dados (termos) e os relacionamentos entre eles (fatos). Já os controles (restrições) e as influências (derivações e regras habilitadoras de ação) compõem o modelo dinâmico onde verificamos os aspectos comportamentais do modelo de negócios. Ao longo das próximas seções observaremos que nos ambientes legados nem sempre esta separação entre modelo estático e dinâmico é clara. Encontraremos muitos fatos inseridos no contexto dos programas (teoricamente pertencentes ao modelo dinâmico) e concomitantemente muitas restrições (*triggers*) deduzidas a partir do modelo estático.

Regras de negócio também podem ser consideradas como metadados. Nesta abordagem as regras são capturadas como elementos de um modelo lógico

permitindo uma boa compreensão das regras, documentação consistente e a construção de modelos conceituais e de dados plenamente compatíveis com as regras. Como vantagens adicionais do tratamento de regras de negócio como metadados podemos elencar a máxima flexibilização, redução do esforço de manutenção (as regras podem mudar com baixo impacto na implementação) e simplificação no projeto e na implementação dos sistemas computacionais [PERK02].

2.2.3 Histórico

Desde os anos 60, as regras foram muito utilizadas pela comunidade de Inteligência Artificial como uma maneira de representar o conhecimento. Na década de 70, surgiram os sistemas especialistas como uma tentativa de representação e implementação de regras [GOTT00].

A partir da década de 80, graças à engenharia da informação, começou a existir uma maior preocupação com a organização e a estrutura dos dados, os quais passaram a representar a empresa como um todo. Houve nesta época a grande explosão dos SGBD relacionais e a forte influência da área de administração de dados nos departamentos de Informática [HAY99, SIQU00].

No final da década de 80, [ZACH87] propõe um *framework* cuja preocupação principal é lidar com as diferentes visões dos participantes do desenvolvimento de um sistema e com a comunicação entre eles. Neste *framework* são identificadas seis diferentes perspectivas (ou fatores básicos) relacionadas em uma matriz com os 5W e 1H (*what, where, who, when, why e how*) apresentadas no Quadro 2-1 [ROSS02].

As RN referem-se exatamente à coluna ***why***, que é uma forma de sensibilizar os projetistas de software sobre os planos e objetivos estratégicos do negócio. Avaliando mais detidamente esta coluna, percebemos que as RN são na

verdade derivadas das políticas de negócio (missão, estratégia) formuladas para refletir influências (tendências), aproveitar oportunidades e considerar riscos e ameaças. As restrições de negócio podem ainda ser consideradas internas (objetivos corporativos, cultura organizacional, regras impostas pela alta administração) ou externas (mercado, leis, comunidade) e representam nesta segunda linha da tabela a perspectiva que a alta gerência tem das RN. A análise (mapeamento dos termos de negócio para as estruturas de dados) está no Modelo de RN (terceira linha), enquanto a decisão sobre a implementação da RN (COBOL, Java, C++, etc.) é refletida no Projeto de RN (linha 4) [ZACH00].

Neste *framework*, as RN ganham maior visibilidade e passam a ser apresentadas como uma derivação das políticas de negócio (missão, estratégia) formuladas para refletir influências (tendências), aproveitar oportunidades e considerar riscos e ameaças. [HAY99, HAY01, LAM00].

	Dados (Que)	**Transform. (Como)**	**Localização/ Conectivida. (Onde)**	**Pessoas/ Colaboração (Quem)**	**Tempo (Quando)**	**Motivação (Por que)**
Objetivos/ Escopo	O que é importante para a empresa	Processos que a empresa realiza	Localidades onde a empresa opera	unidades organizacionais	Eventos/ciclos de negócio	Objetivos/Estratégias de negócio
Modelo do Negócio	Modelo E-R	Modelo de Processos de negócio	Nós e links da rede	Fluxograma com papéis, habilidades e segurança.	Prioridades para o negócio	Restrições (constraints) do negócio
Modelo dos Sistemas de Informação	Modelo de dados totalmente normalizado	Diagrama de fluxo de dados essencial; arquitetura da aplicação	Arquitetura de sistemas distribuídos	Interface humana (papéis, dados, acessos)	Diagrama de dependência, história da vida da entidade (estrutura do processo)	Modelo de RN
Modelo de Tecnologia	Arquitetura de dados; mapeamento dos dados legados	Projeto do sistema; estrutura, diagrama, pseudo-código	Arquitetura do sistema (hardware, tipos de software)	Interface do usuário (como o sistema irá se comportar); projeto de segurança	Diagrama de Fluxo de Controle (Estrutura de controle)	Projeto de RN
Representações Detalhadas (Implement.)	Projeto de dados (desnorm), projeto de armazen. Físico	Projeto detalhado de programas	Arquitetura de rede	Telas, arquitetura de segurança	Definições de tempo	Especificação de regras na lógica dos de programas
Funções do sistema	Dados convertidos	Programas executáveis	Recursos de comunic.	Pessoal treinado	Eventos de negócio	Regras definidas

Quadro 2-1 - O framework de Zachman com a sexta coluna assinalada, onde as RN são explicitamente relacionadas (Fonte: [ZACH00]).

Na década de 90, com o amadurecimento dos conceitos de orientação a objeto, observou-se que na metodologia tradicional de desenvolvimento de sistemas, muito da essência do negócio ainda não era conhecida e, desta forma, constatou-se a existência de um outro elemento de conhecimento que seria crítico para o entendimento da natureza de uma organização: as RN. [HAY99]

2.2.4 Classificação de RN

Em [SULA00] encontramos uma série de menções a propostas de classificação de RN, desde a abordagem de [ROSS97] levando em conta o modelo conceitual e uma comparação com a tabela periódica de elementos, até a classificação proposta por [DATE00a], considerando o modelo relacional de dados. Há tentativas brasileiras, como pode ser verificado em [SULA00] através da Classificação de Tipos Básicos de RN (CTBRN), implementada no produto G-REN (Gerador de RN) desenvolvido nas instalações do SERPRO [SULA99]. Contudo, dentre as classificações existentes, a mais consagrada encontra-se em [GUID97] onde são definidos os seguintes tipos fundamentais de RN:

> **Termos** - Relacionados ao modelo de dados, detalhando um aspecto específico e estático do negócio. Termos são os principais conceitos e definições relacionados ao negócio encontrados em documentos não padronizados, glossários ou representados nas entidades de um modelo de dados [SULA00].
> ROSA – Exemplo das regras
>
> **Fatos -** Os fatos representam as relações entre as entidades ou entre estas e seus atributos, descrevendo a natureza ou estrutura operacional do negócio na forma de sentenças em linguagem natural ou como relacionamentos, atributos e estruturas de generalização em modelos gráficos [HAY99, SULA00].

Derivações - São cálculos matemáticos ou mudanças de estado realizadas sobre termos, fatos ou mesmo outras derivações [DATE00a, HAY99] Em última análise determinam como um conhecimento ou informação pode ser transformado em outro [SULA00]. Apesar de facilitarem a compreensão do modelo, as regras de derivação costumam não ser encontradas nos modelos relacionais por ferirem as regras de normalização, já que a informação derivada pode ser obtida a partir de outras colunas de tabela.

Restrições – Restringem algum aspecto do negócio, que pode estar relacionado tanto ao modelo estático (restrições de integridade) quanto ao modelo dinâmico (ações tomadas pela organização, indicando direções obrigatórias ou adequadas para tratamento da informação) [HAY99].

Regras Habilitadoras de Ação - Considerando a ação como um conjunto de regras, encontramos aqui a inferência condicional, cujo resultado pode ser falso ou verdadeiro. A partir deste resultado, um conjunto de regras ou ações passa a ser executado.

2.2.5 Produtos Comerciais

O trabalho considerado como o mais completo sobre RN é [ROSS97], cujo autor é tido por muitos como o pai da abordagem de RN. Esta obra foi criticada por ser excessivamente teórica e por não apresentar uma proposta de arquitetura para implementação. Pouco tempo depois, o autor lança uma revisão mais condensada de forma a facilitar sua aplicação prática [ROSS98]. Mais recentemente, [DATE00a, DATE00b, DATE00c] descreve uma abordagem complementar sobre RN tendo como ponto de partida o modelo relacional como base para estudo das RN. Outra

autora de destaque na área é [HALL01a, HALL01b], que propõe uma metodologia passo-a-passo desde a fase de definição de escopo até a especificação, automação e gerenciamento de regras. A metodologia consiste de fases (*Phases*), tarefas (*Tasks*) e técnicas (*Techniques*) para gerenciamento de RN desde sua captura inicial até automação e mudança. A título de ilustração segue o Quadro 2-2 com os principais produtos envolvendo RN.

Ferramenta	Fornecedor	Descrição
Aion	Platinum	Considerado um gerador de aplicações baseado em sistemas especialistas, o Aion é fundamentado em regras para desenvolvimento e implementação de componentes inteligentes combinando RN e programação OO para criação e manutenção de aplicativos.
BlazeAdvisor	Brokat	Esta ferramenta faz parte da solução integrada Blaze (sucessora da NeuronData e apresenta recursos para manutenção de RN durante a operação da empresa (*on the fly*) usando uma combinação de linguagem "quase natural" e *templates* direcionados para os profissionais do negócio.
BRSRuleTrack	BRSolutions	Ferramenta automática para registro, organização e gerenciamento de RN tendo como base uma perspectiva puramente de negócio, independente de peculiaridades dos ambientes envolvidos.
HOW	Riverton Software	HOW é uma família de produtos para desenvolvimento/implantação baseado em componentes. Sua ferramenta de modelagem suporta análise e projeto orientados a negócio e é capaz de derivar RN a partir de casos de uso (UML), ligando as regras aos objetos de negócio que as implementam.
ILOG Rules	ILOG	Máquina de regras que permite às aplicações incorporar regras dinâmicas para controlar fluxos de dados em tempo real.
Usoft	Usoft Developer	Gerador de aplicações baseado em um repositório, fortemente orientado para SQL
Versata	Versata	Sucessor do Vision Builder, é um gerador de código baseado em regras declarativas

Quadro 2-2 - Principais produtos comerciais que tratam sobre RN

Na área de representação de regras de negócio convém destacar o projeto Atenas, desenvolvido pela COPPE/UFRJ em parceria com a Marinha do Brasil. [ZIMB01] Seu objetivo é servir como uma base formal para a documentação das regras de negócio de um sistema de informação, as quais se dividem em três grupos: extraídas automaticamente do esquema do banco de dados, cadastradas manualmente pelos usuários e inferidas automaticamente a partir da base de conhecimento. Em relação à nossa abordagem, as funcionalidades do Atenas

atendem a objetivos um pouco distintos. A leitura de um esquema relacional não se aplica ao ambiente do **Source Inspector**. O cadastramento manual implica na existência de um editor de regras que não é o foco do nosso projeto. A inferência automática de regras também é uma abordagem que vai além do nosso escopo, visto que os recursos disponíveis no **Source Inspector** neste sentido dependem muito da intervenção do especialista. Diante do exposto, vemos o Atenas como uma ferramenta bem mais genérica, mas que não aborda o problema do código COBOL propriamente dito, ainda mais no ambiente UNISYS/DMSII.

2.2.6 Identificação e extração de regras de negócio

A complexidade das RN pode variar desde simples verificações até complicadas rotinas estatísticas. Além disso, elas podem ser específicas de uma empresa, de uma área ou ramo de negócio, ou ainda aplicáveis a todas as empresas atuando em determinado país ou região [ROSS97]. Finalmente, as RN estão em constante mutação, acompanhando o movimento dos negócios de uma forma geral. [SIQU00, SEEC00].

Se a identificação das regras por si só já não é uma tarefa fácil, sua extração a partir de um código legado torna-se um problema maior ainda sem um método que contemple preferncialmente tanto os dados quanto o fluxo da aplicação. Este processo pode ser dividido em duas fases. Na primeira delas o principal trabalho é de organização e pesquisa sobre o software legado. Em seguida, num processo interativo, as RN podem ser encontradas, classificadas e catalogadas de tal forma a promover a independência do conhecimento sobre o negócio [SULA99].

O mapeamento de objetos em sistemas legados requer a presença de um especialista capaz de identificar objetos próximos ao domínio da aplicação e determinar alguma relação entre eles. Se a análise dos dados de um sistema legado ainda pode ser auxiliada por ferramentas de mercado (como o Evoke Axio, por

exemplo), os procedimentos dos códigos legados geralmente são tão complexos que fica praticamente impossível automatizá-los. Neste aspecto convém considerar que qualquer solução apresentada não pode ser tão ambiciosa a ponto de extrair a totalidade das regras implementadas, especialmente porque mesmo aplicando recursos de mineração de regras [LAM02].

Uma das abordagens mais utilizadas para tratamento de sistemas legados é utilizar um extrator que realiza análises léxicas, sintáticas e de fluxo de dados sobre os códigos fonte e armazena referências num repositório. Posteriormente, uma ferramenta de consulta é utilizada sobre o repositório com o objetivo principal de estabelecer uma filtragem que resultará nos itens mais relevantes para a compreensão do sistema, mais especificamente, os termos e as regras de negócio. Finalmente, uma ferramenta de visualização permitirá a interação entre o especialista de domínio e o ambiente legado. As tarefas de extração-consulta-visualização são continuamente repetidas até que se tenha uma idéia satisfatória do sistema legado. Naturalmente, ferramentas tão poderosas não se restringem apenas à identificação de objetos, sendo bastante úteis para geração de documentação e treinamento de novos técnicos no ambiente da organização [SEEC00].

2.3 Trabalhos correlatos

No âmbito da Engenharia de Software, identificamos uma série de ferramentas e estudos aplicados ao código legado, especialmente quando este é representado em COBOL. A maioria delas entretanto, está particularmente interessada na manutenção, transformação ou migração de sistemas, sem uma preocupação específica com a extração de RN: [FREI97, LEIT95, PAUL99, PROG99, SAYL96, SELL97, STRO99, TONE00].

Uma das abordagens mais interessantes no tratamento específico para COBOL foi desenvolvido pelo Grupo de Pesquisa em Programação da Universidade

de Amsterdam. Em COBOL não é possível separar definições de tipo das declarações de variáveis. A partir desta constatação, foi desenvolvida uma infraestrutura técnica baseada no meta-ambiente ASF+SDF, uma combinação de dois formalismos de especificação: o de Especificação Algébrica (***A****lgebric* ***S****pecification* ***F****ormalism*) e o de Definição de Sintaxe (***S****yntax* ***D****efinition* ***F****ormalism*). Dentre os aspectos mais interessantes deste trabalho está o esforço para identificação de objetos em sistemas legados de tal forma a estabelecer uma transformação para componentes levando em conta fatores chaves para a qualidade de software (correção, robustez, extensibilidade e reusabilidade) [BRAN97, DEUR99, SIQU00].

As propostas existentes para tratamento de RN são ainda tímidas em termos de mercado devido a sua complexidade e uma forte necessidade de interação com os analistas do negócio. O esquema, de uma forma geral, segue o que está apresentado na Figura 2.3. Os objetos legados (programas, dados, *jobs*) são inseridos em um repositório unificado. A partir deste repositório, por algum método de extração, tanto o modelo de dados quanto o de processos é gerado em alguma forma particular de representação. A partir deste ponto ocorrem as propostas para identificação e extração de regras de negócio com a interferência de um especialista de domínio. O resultado final é um modelo de negócios. [HEIM00]

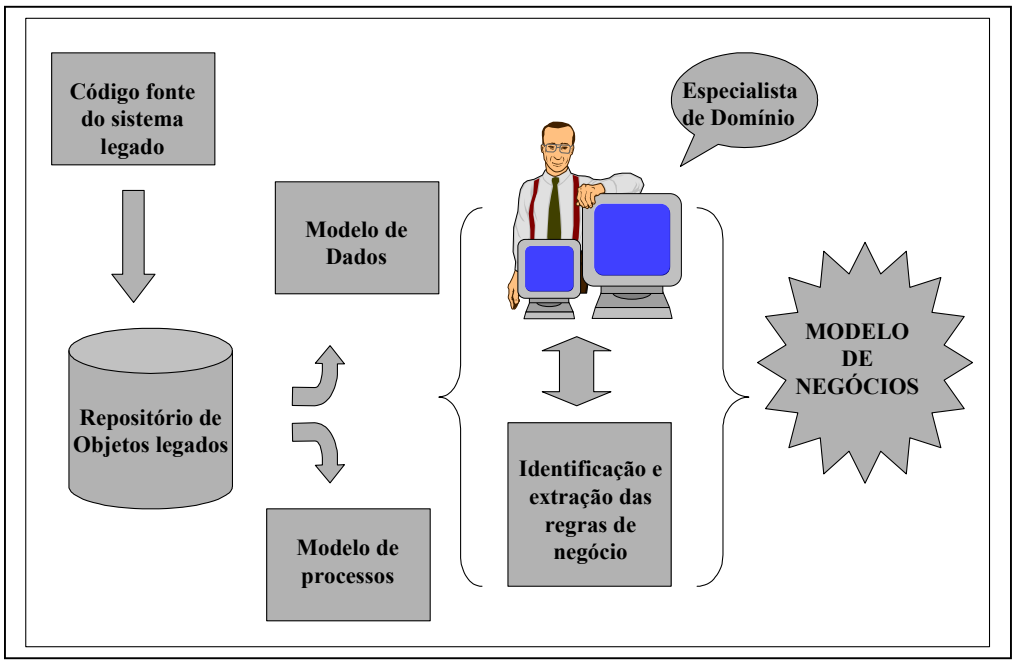

Figura 2-2 - Extração de modelo de negócios a partir de sistemas legados (Fonte: [HEIM00] apud [SIQU00]).

Seguindo esta metodologia de extração, encontramos as seguintes iniciativas:

MOSAIC/SEEC - Em [HEIM00] encontramos uma metodologia com etapas bem definidas apresentando inclusive uma classificação elementar para a catalogação das RN obtidas a partir da utilização de suas ferramentas. O MOSAIC/SEEC já apresentou comprovada eficiência na modernização de sistemas, mas o mesmo ainda não pode ser dito sobre o tratamento das regras de negócio.

ASSET MINER – Este produto da Microfocus é um componente introduzido no Revolve, uma ferramenta mais completa de manutenção. O Asset Miner permite que os comandos que

manipulam dados no COBOL (READ, WRITE, LOCK, MOVE, etc.) recebam um peso conforme sua possibilidade de extração de regras nas sentenças do programa. O parâmetro peso é introduzido pelo usuário [NETR00, SAYL96].

DPUTE - Em [HUAN98] é apresentado o conjunto de ferramentas DPUTE (*Data-Centered Program Understanding Tool*) que possui o objetivo de auxiliar a compreensão e permitir a extração de RN de programas COBOL. A solução apresentada combina diversas técnicas, como classificação de variáveis, heurísticas para identificar critérios de *slicing*, representações múltiplas de RN, análise de fluxo de dados *bottom-up* e abstração hierárquica. A abordagem utilizada é centrada em dados e nas suas relações. A implementação desta solução através do DPUTE foi desenvolvida tanto para ambiente Unix como para OS/2 e segue princípios bastante semelhantes aos aplicados em [GAVI00], apresentando porém profundidade maior na extração de RN nos programas propriamente ditos, utilizando técnicas mais avançadas de *slicing*.

MINEIT – Este produto da Intercomp, Inc. segue princípios semelhantes aos descritos para as soluções anteriores. Voltado para o ambiente CICS (*Customer Information Control System*) da IBM. Seu núcleo é composto por um pacote de compiladores que conseguem interpretar várias derivações do COBOL além de comandos SQL e gerar no final do processo componentes em Java contendo as RN [INTE00].

G-REN – G-REN é o acrônimo de Gerador de RN, um produto totalmente desenvolvido no SERPRO. Visa integrar as atividades dos Administradores de Dados, responsáveis pela especificação das RN junto aos gestores de sistemas de informação do cliente, e dos Administradores de Banco de Dados (DBA), responsáveis pela implementação destas regras. A partir dsta integração o G-REN busca uma maior qualidade de especificação das regras e, consequentemente, um aumento de produtividade por conta da geração automática das regras [SULA00].

3 HEURÍSTICAS E O PROCESSO PARA IDENTIFICAÇÃO DE REGRAS DE NEGÓCIO

"Eu só sei que eu tenho de
apertar este botão aqui,
o resto o sistema faz."

Usuário de um poderoso sistema legado.

"Eu não sei como isto funciona aqui na empresa.
Pergunte ao analista."

Depoimento comum em organizações
onde as regras de negócio fugiram
ao controle dos especialistas de domínio.

Neste capítulo descreveremos o processo adotado para extração das RN, apresentando as heurísticas definidas para este fim. Como se trata de uma solução específica para um determinado parque computacional, faremos uma breve descrição do ambiente UNISYS. Muitas das heurísticas foram definidas levando-se em conta esta característica básica.

3.1 Um metamodelo DMSII/COBOL

3.1.1 Um metamodelo para os bancos de dados DMSII

O banco de dados DMSII é um sistema proprietário de armazenamento hierárquico do ambiente UNISYS e cuja linguagem utilizada para definir suas estruturas denomina-se DASDL (*Data And Structure Definition Language*). Esta linguagem possui uma estrutura de formato rígido, facilitando a definição de um processo de extração de informações a partir dos arquivos contendo código fonte. Cada arquivo DASDL representa o esquema de um único banco de dados e a Figura 3.1 mostra graficamente a relação entre os diversos elementos definidos pelo DASDL [UNIS97b].

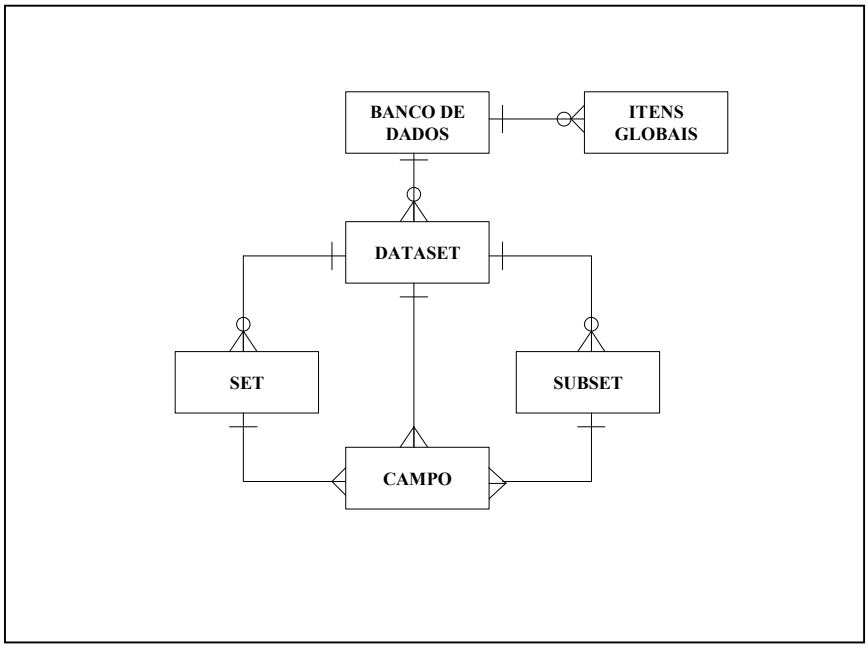

Figura 3-1 - Diagrama da estrutura do DMSII (Fonte: [GAVI00]).

O Quadro 3-1 explica com mais detalhes cada um dos componentes DASDL.

Estrutura	Descrição
BANCO DE DADOS	Coleção centralizada de dados armazenados em um ou mais arquivos, podendo ser acessada simultaneamente por vários programas aplicativos. Não está relacionado a nenhum sistema específico e deve conter pelo menos um DATASET.
CAMPO	Item de dado pertencente a um DATASET. Pode estar associado a quaisquer SETs ou SUBSETs do DATASET
DADOS GLOBAIS	Itens de dados que não sejam parte de algum DATASET são chamados itens de dados globais, que geralmente consistem de informações como totais de controle, datas do sistema e outros tipos de informação aplicáveis a todo o banco de dados.
DATASET	Coleção de um ou mais itens de dados (campos) armazenados em um arquivo e que pertence a um único banco de dados.
SET	É uma estrutura que permite acesso a todos os registros de um DATASET a partir de alguma seqüência lógica, contendo uma entrada para cada registro no DATASET. Se itens chave forem especificados para o SET, os registros no DATASET serão acessados pela chave especificada. Caso contrário o acesso aos registros ocorre seqüencialmente. Vários SETs podem ser declarados para um único DATASET permitindo que os dados possam ser acessados a partir de diversas seqüências. O conceito de SET pode ser mais bem entendido através do catálogo do modelo em redes [ELMA94, DATE00b], mas estabelecendo um paralelo com o modelo relacional, trata-se de uma projeção (vertical) ou *view* sobre o conjunto original de dados.
SUBSET	Enquanto o SET determina uma projeção, o SUBSET determina uma seleção (horizontal) ou *constraint*. Um SUBSET é uma estrutura semelhante ao SET, com a ressalva de que um SUBSET precisa apenas ser referir aos registros selecionados no DATASET e esta seleção possui uma condição lógica associada.

Quadro 3-1 - Descrição dos principais elementos utilizados pelo DASDL (Fonte: [GAVI00]).

3.1.2 Um metamodelo dos programas COBOL

Conforme vimos no capítulo 2, mesmo com o advento de novas linguagens e técnicas de programação, COBOL permanece como uma ferramenta muito presente nos sistemas legados das organizações, especialmente aquelas inseridas em um contexto de negócios [SELL97, TONE00, VERH00].

Para muitas aplicações em COBOL, os dados representam o coração do sistema. Por isso, os registros de dados são utilizados como ponto de partida para muitas abordagens de identificação de objetos. Além disso, o modelo de dados é notadamente mais estável que o de funções, sendo a análise do primeiro bem mais fácil [HUAN98].

A sintaxe do COBOL possui um formato rígido. A análise da gramática nos permitiu identificar um conjunto de entidades a partir das quais é possível realizar extração de informações. A Figura 3-2 ilustra estas entidades, enquanto o Quadro 3-2 nos fornece uma explicação mais detalhada de cada um dos componentes COBOL.

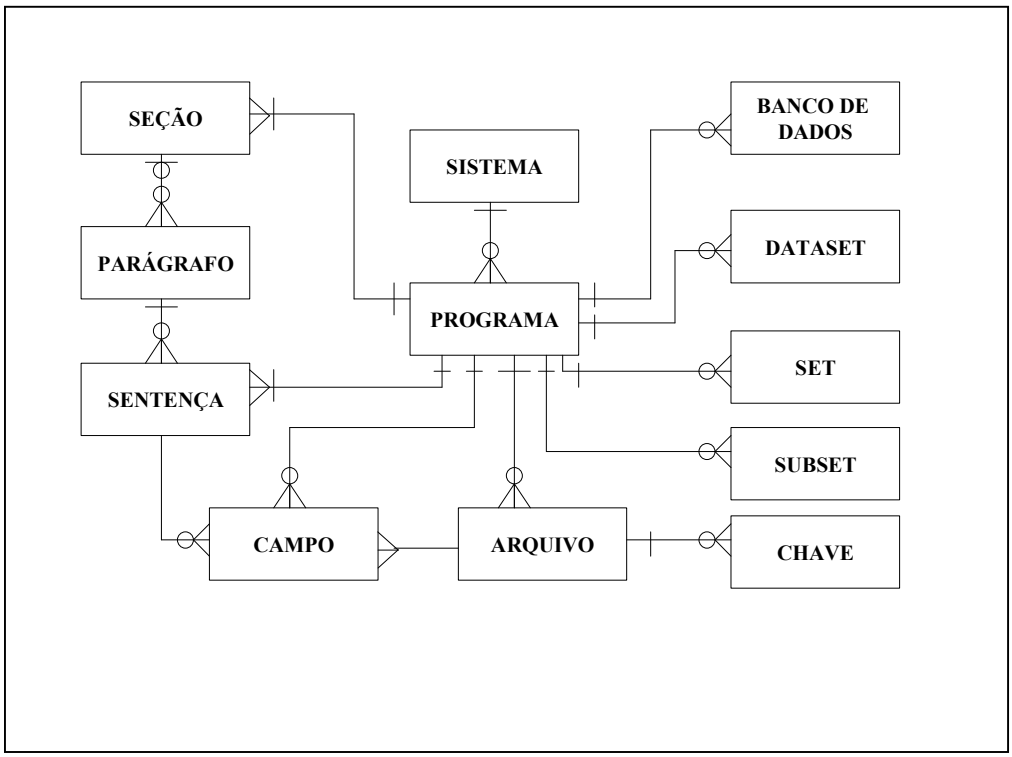

Figura 3-2- Diagrama da estrutura de um programa COBOL.(Fonte: [GAVI00]) .

Estrutura	Descrição
SISTEMA	Grupo de programas que podem acessar quaisquer banco de dados.
PROGRAMA	Conjunto de declarações e sentenças pertencentes a um único sistema. Pode acessar qualquer banco de dados e, neste banco, os DATASETs, SETs e SUBSETs que desejar. Não há restrição quanto ao número de arquivos, nem variáveis de memória.
ARQUIVO	Coleção de pelo menos um campo. Cada arquivo físico pode ser referenciado por vários programas, mas sua definição lógica é específica para um programa. Se for indexado, sua chave será armazenada.
CAMPO	Item elementar de dado. Pode ser um campo de arquivo ou uma variável de memória declarada na WORKING-STORAGE SECTION.
SEÇÃO	Compartimento do COBOL que pode armazenar parágrafos ou sentenças isoladamente.
PARÁGRAFO	Estrutura semelhante à SEÇÃO, armazenando apenas sentenças. Estamos considerando apenas os parágrafos existentes na PROCEDURE DIVISION.
SENTENÇA	Representa um conjunto de comandos COBOL terminado por ponto, e pode acessar uma série de campos para consulta ou atualização. As sentenças que não acessam campo algum não são tratadas ou armazenadas.

Quadro 3-2 - Descrição dos principais elementos utilizados pelo COBOL (Fonte: [GAVI00]).

A noção do ambiente UNISYS descrita acima nos permitirá uma melhor compreensão das heurísticas, conforme veremos a seguir.

3.2 Heurística e Tipos de RN identificadas

Heurística pode apresentar diversos significados, como palpites, regras baseadas na experiência, julgamentos intuitivos ou simplesmente senso comum. Uma definição mais formal estabelece que as heurísticas representam estratégias montadas a partir de informações incompletas com o objetivo de solucionar algum problema. O caminho escolhido através das heurísticas pode não ser a melhor solução, mas muito provavelmente representará uma das melhores e com grande chance de ser factível computacionalmente falando [PEAR85].

Juntamente com as heurísticas, para identificar as RN utilizaremos um resumo da classificação existente em [GUID97]. Trabalharemos com cinco categorias de RN (*Termos, Fatos, Restrições, Derivações e Regras Habilitadoras de Ação (RHA)*). Os *Termos* exercem importante papel na identificação das demais regras. A partir de um BD identificamos os termos como variáveis globais de BD, nomes de DATASETs e atributos de DATASETs. Além disso, podemos identificar os termos também no código COBOL em arquivos e tabelas definidas para os programas. A identificação de termos pressupõe um processo que será analisado com mais detalhes adiante.

Ainda analisando o sistema legado no ambiente UNISYS/COBOL/DMSII, consideraremos que as RN podem ser divididas em dois grupos: as RN obtidas a partir dos itens de dados declarados nos bancos de dados (denominadas simplesmente **RegrasBD**) e as RN obtidas a partir dos programas (denominadas **RegrasProg**). Estas, por sua vez, ainda se subdividem em **FatosProg** (derivados das declarações de arquivos e itens de dado dos programas) e **RegrasSentença** (sentenças de comando existentes na PROCEDURE DIVISION dos programas COBOL) [SIQU02].

As **RegrasBD** podem ser de três categorias: *Fatos*, *Derivações* e *Restrições*. São considerados *Fatos* as RN que estabelecem relacionamentos entre os itens de dados identificados (p. ex. entre DATASETs, entre um DATASET e seus atributos), mostrando como os termos se relacionam. Através da leitura do banco de dados é possível levantar os fatos e os DATASETs envolvidos (explicitamente referenciados). Identificaremos basicamente quatro tipos de fatos de negócio: Associação, Ordenação, Herança e Dependência. Quase todos os fatos derivam do modelo de dados. Apenas alguns tipos de Associação envolvendo os arquivos de programa constituem os **FatosProg**, como veremos mais adiante.

São consideradas *Derivações* as RN obtidas a partir dos comandos habilitados pela cláusula AGGREGATE do DASDL, ou as iniciações definidas para os termos de negócio e que não sejam as padronizadas pela cláusula DEFAULTS. Finalmente, as *Restrições* são obtidas a partir das cláusulas VERIFY (para DATASET) e WHERE (para SUBSET), às quais são relacionadas expressões lógicas.

As **RegrasSentença** também podem ser de três categorias: *Derivações, Restrições* e *Regras Habilitadoras de Ação*.

Devido à incapacidade do DMSII armazenar estruturas mais elaboradas de bancos relacionais, tais como *stored procedures*, quase todas as derivações e cálculos de um sistema legado UNISYS são identificadas nos fontes de programa (no caso específico da nossa análise, fontes COBOL).

Além dos comandos aritméticos, o comando mais importante para identificação de derivações é o MOVE. No **Source Inspector**, este comando tratado basicamente de duas formas:

- Como uma declaração onde um identificador (ou grupo de identificadores) é consultado e outro identificador (ou grupo de identificadores) é atualizado.

- Como uma declaração onde um valor atualiza um identificador ou grupo de identificadores.

Diante dos pressupostos acima, definimos cinco tipos básicos de derivação:

- Simples – Movimentação entre campos de um mesmo domínio quando um deles for caracterizado como termo de negócio;
- Mudanças de estado – Qualquer atribuição de um valor a um item de dado;
- Mudanças de domínio – Caracterizam movimentação de valores entre itens de dados com diferentes tipos ou tamanhos.
- Cálculos – Qualquer resultado de comandos ou expressões aritméticas em declarações imperativas da linguagem;
- Iniciações – Um caso especial de mudança de estado onde são utilizadas algumas palavras-chave do COBOL.

As *Restrições* são comandos condicionais obtidos a partir de certos parágrafos e as *Regras Habilitadoras de Ação* são comandos condicionais fazem inferências ou determinam ações diretas (geralmente comandos do tipo PERFORM). Enquadram-se nesta categoria, elementos fundamentais da linguagem COBOL, tais como os comandos condicionais simples (IF) e comandos condicionais com iteração (PERFORM UNTIL).

Veremos com mais detalhe como esta identificação é realizada a partir da análise das heurísticas. A Figura 3-3 abaixo fornece uma ilustração do esquema de hierarquia das regras.

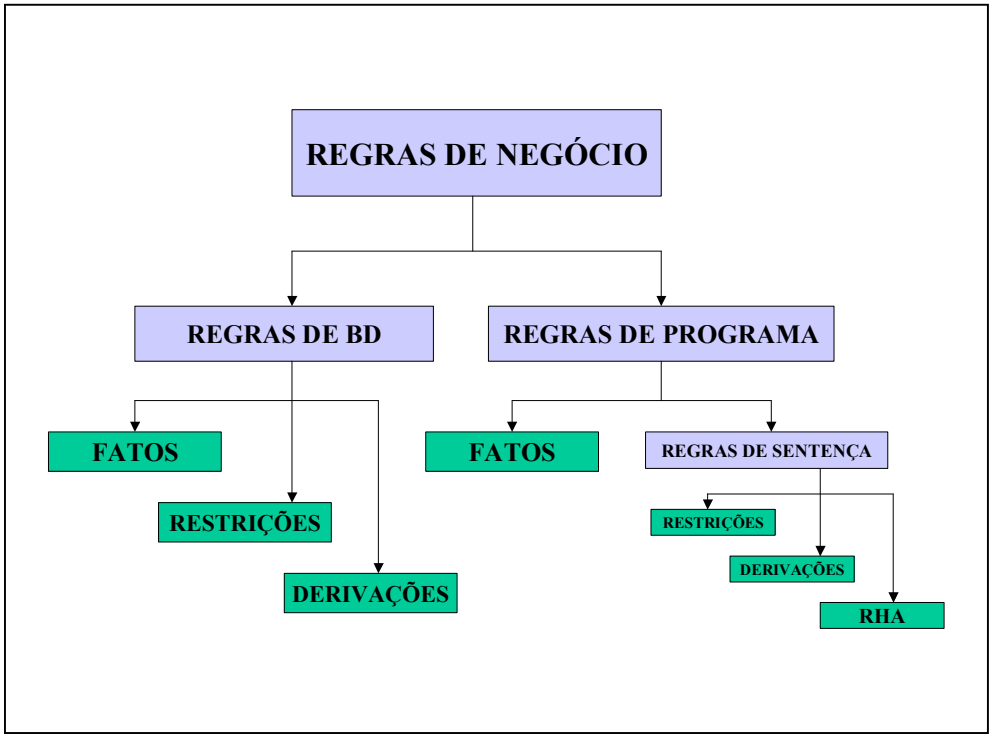

Figura 3-3- Hierarquia de RN na extração dos fontes UNISYS.Fonte:[SIQU02]).

3.3 O processo de Identificação de Termos de Negócio

O maior desafio na identificação dos *termos* em ambientes legados é estabelecer um processo de tal forma que, a partir dos *termos* seja possível obter as demais RN. Nossa proposta define um processo em quatro fases: Fase 1 – Preparação dos fontes, Fase 2 – Cadastramento de parâmetros, Fase 3 – Extração das regras de BD e Fase 4 – Extração das regras de programa. Estas fases estão representadas, respectivamente, nas Figuras 3.4 a 3.7.

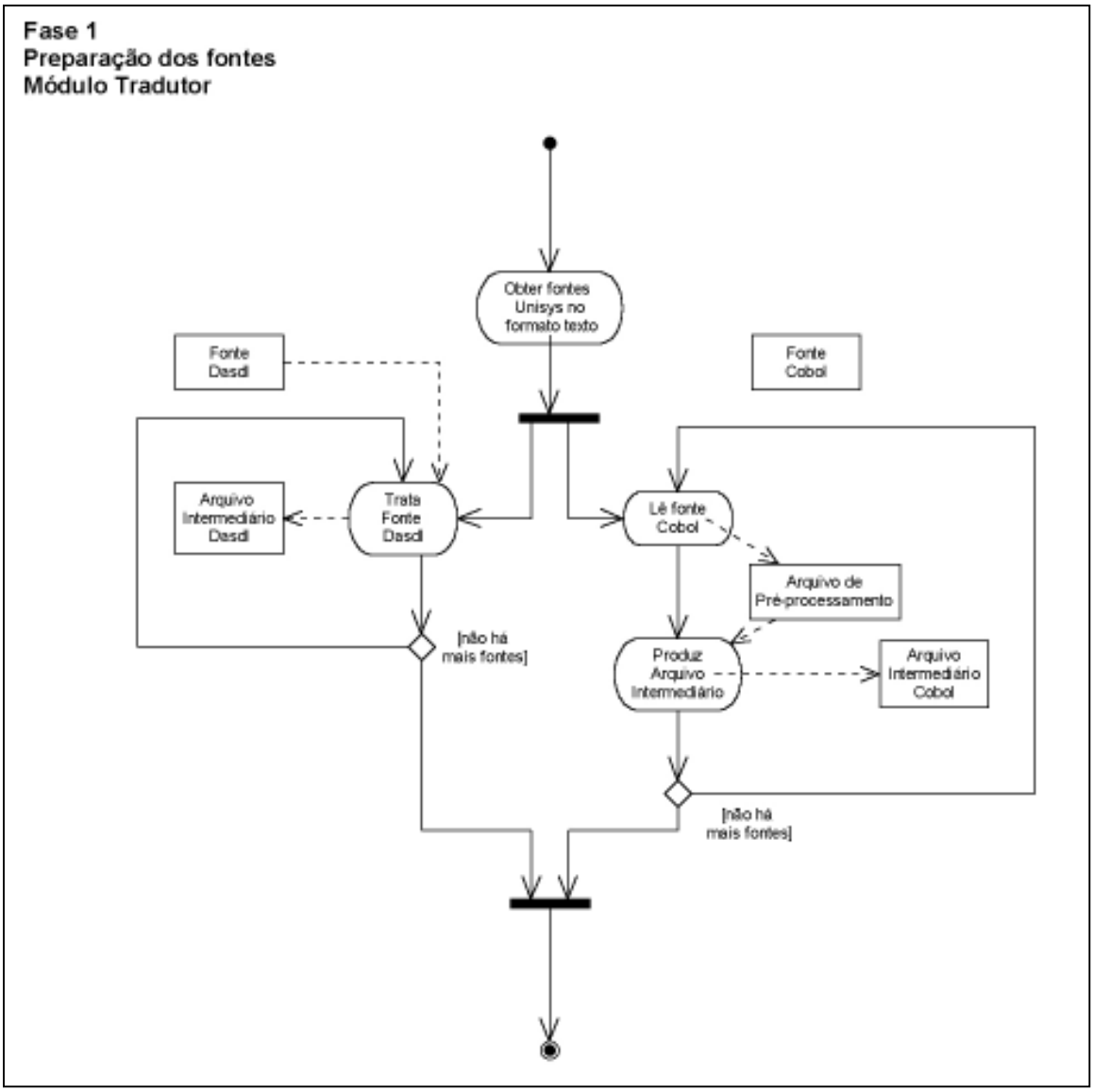

Figura 3-4 - Diagrama de atividades. Fase 1 - Preparação dos fontes.

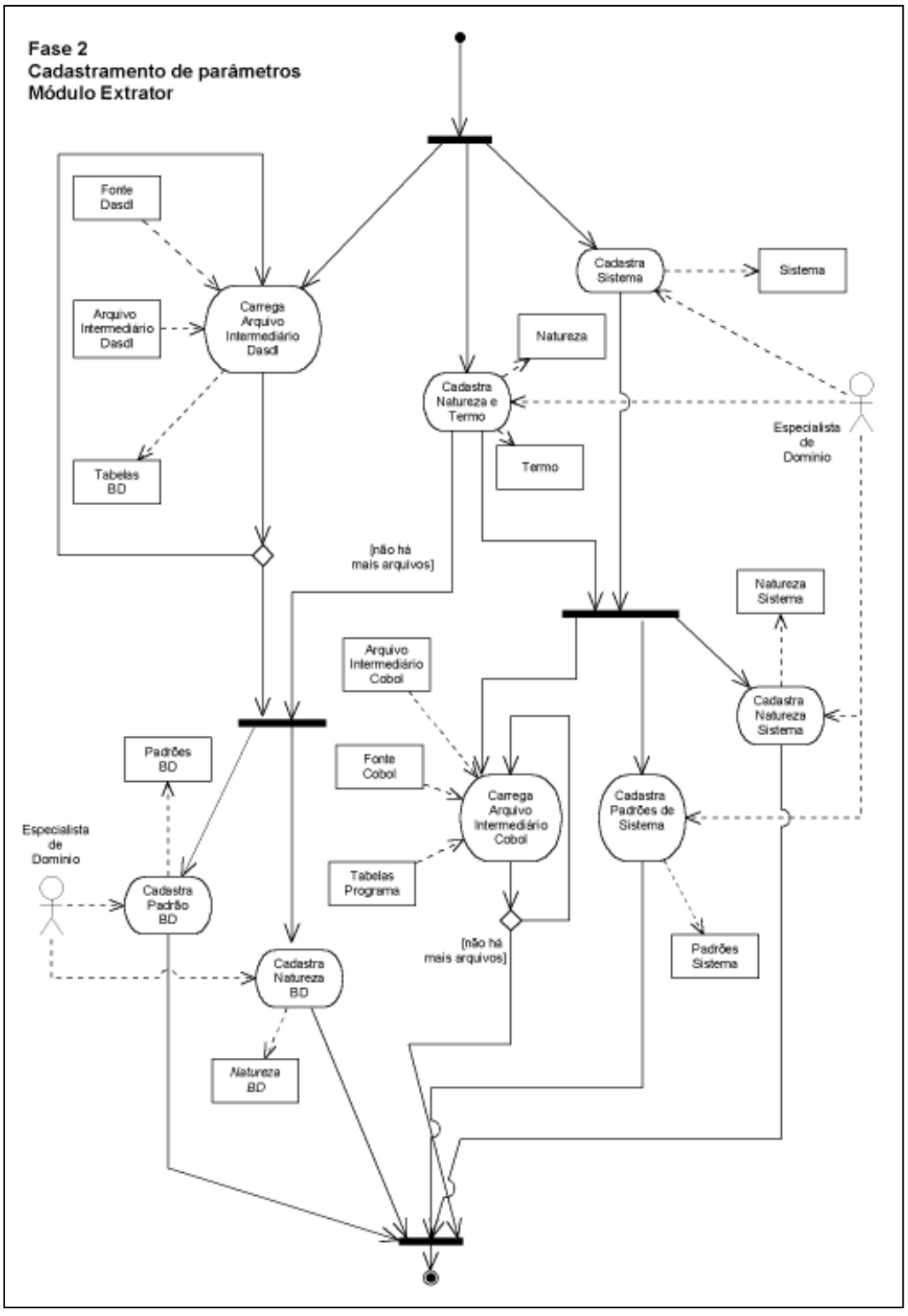

Figura 3-5 - Diagrama de atividades. Fase 2 - Cadastramento de parâmetros.

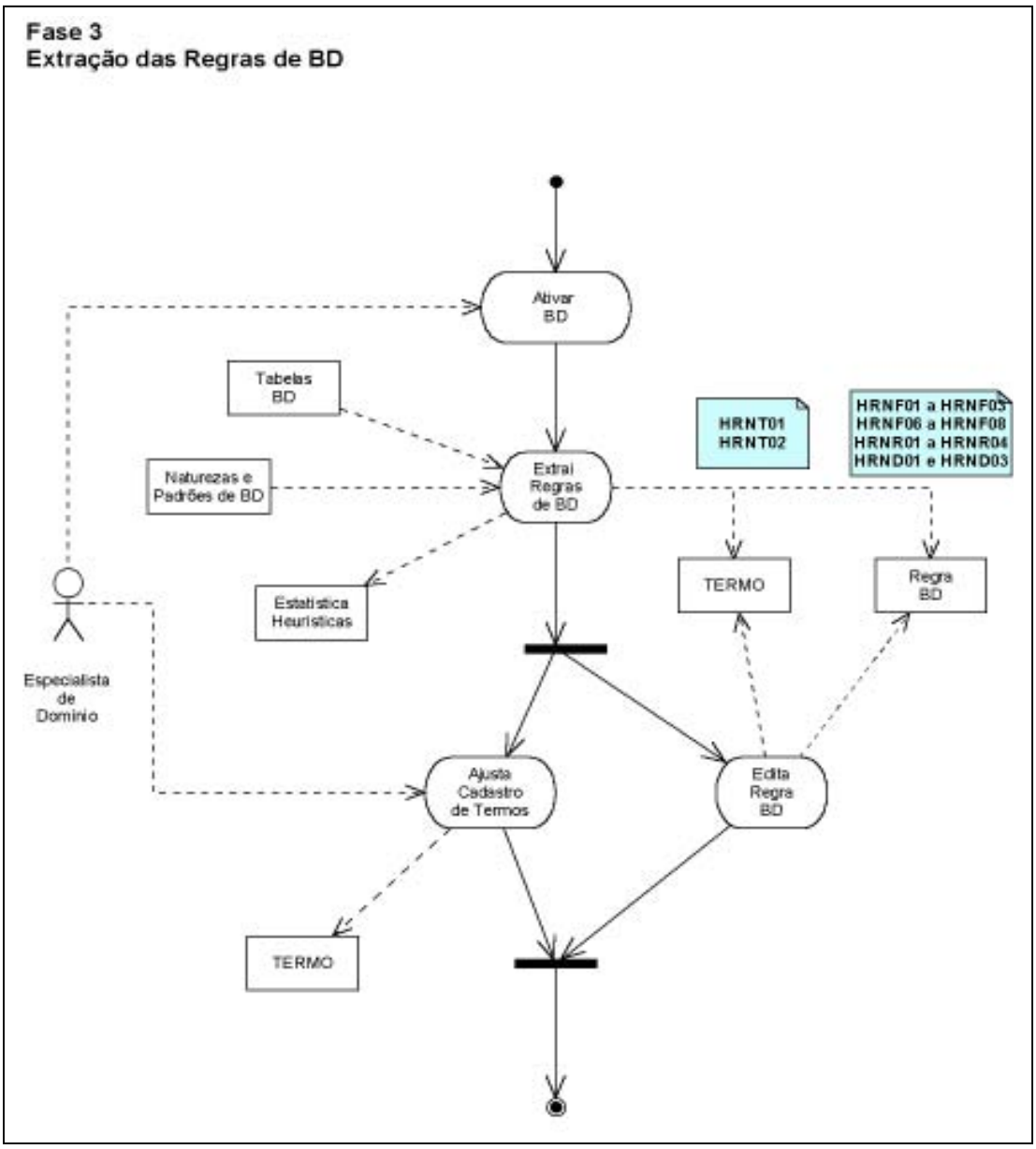

Figura 3-6 - Diagrama de atividades. Fase 3 - Extração de Regras de BD.

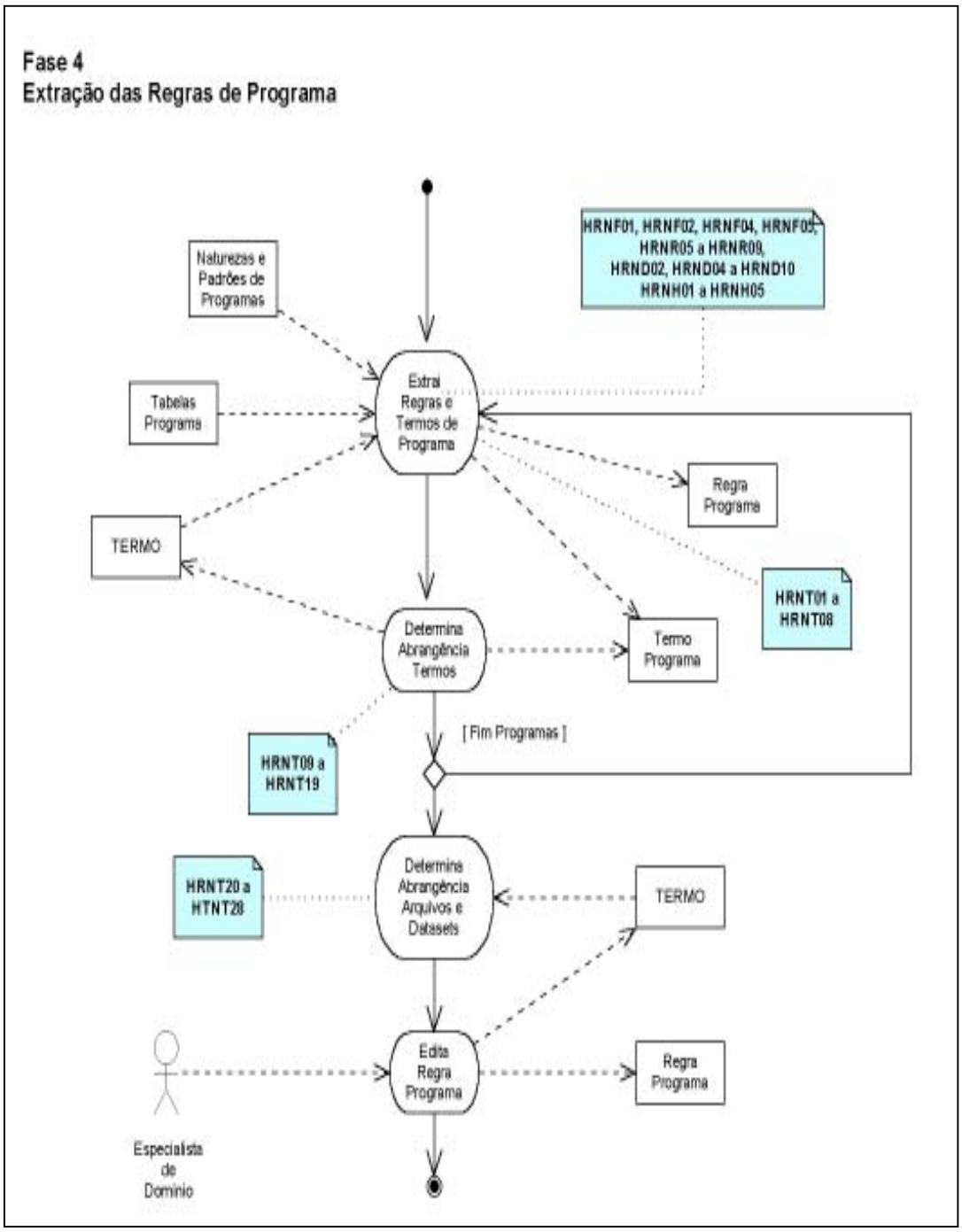

Figura 3-7 - Diagrama de atividades. Fase 4 - Extração de Regras de Programa.

3.3.1 Fase 1 – Preparação dos fontes

3.3.2

Na primeira fase ocorre a preparação dos fontes originais para que eles possam ser tratados pelo Módulo Tradutor do **Source Inspector**, conforme veremos com mais detalhe na descrição da ferramenta.

Apesar de parecer uma atividade muito específica da ferramenta, esta etapa do processo é importante porque vai viabilizar o acesso aos fontes existentes no ambiente UNISYS. Tanto os fontes DASDL como os fontes COBOL devem ser convertidos para um formato texto a fim de que possam ser tratados e convertidos em arquivos intermediários próprios para a ferramenta. No caso de programas COBOL vale destacar ainda a necessidade de um pré-processamento que possui o objetivo de retirar caracteres de controle e registrar a numeração do fonte para posterior identificação das linhas de código.

3.3.3 Fase 2 – Cadastramento de parâmetros

Em seguida, ocorre o cadastramento de parâmetros para extração. Conforme vimos no capítulo 2, a solução consagrada para extração de RN é semi-automática e interativa, requisitando a participação de um especialista de domínio.

Num processo interativo, a participação do especialista de domínio não precisa ocorrer necessariamente após uma primeira avaliação do código. Antes mesmo da análise dos fontes, alguns parâmetros podem ser introduzidos para facilitar o processo de extração. Nossa proposta é que esta intervenção inicial contemple as seguintes atividades:

- Cadastramento de Naturezas e Termos de Negócio.

A maior contribuição para o cadastro de termos de negócio será dada pela ferramenta de extração, mas isso não impede que o especialista de domínio contribua antes mesmo das atividades de extração propriamente ditas. Com isso, quanto mais completo estiver o cadastro, mais precisos serão os resultados. As naturezas, por sua vez identificam um tipo de dado que seja atributo de uma entidade ou item de dado de um programa. Esta tabela facilita a descrição do termo de negócio ao identificar alguma seqüência de caracteres que indique o tipo de informação referenciada (valor, data, texto). Existe um cadastro genérico para naturezas a partir do qual o especialista de domínio seleciona quais as naturezas válidas para cada banco de dados e/ou sistema.

- Cadastramento de Padrões de Nomenclatura

Os padrões têm como objetivo identificar os *Termos* dentro de itens de dados com uma nomenclatura mais extensa. Em muitas instalações os nomes dos atributos das tabelas ou arquivos são compostos por siglas que se referem ao sistema ou ao BD, dependendo do padrão de nomenclatura adotado. A utilização dos padrões facilita a busca de regras em instalações com razoável grau de organização. Identificamos três tipos de padrões:

  - Padrões de BD – Regras aplicáveis na identificação de termos em DATASETs, SETs e campos para cada banco de dados;

  - Padrões de Sistema - Da mesma forma que é possível definir alguns padrões para os BDs, é possível estabelecer padrões para facilitar a identificação de termos de negócio nos programas. Definimos dois tipos básicos:

- Padrões para programas – Identificação específicas para itens de dados e variáveis a fim de facilitar a extração de termos e regras.

- Padrões de parágrafos - O caso da padronização dos parágrafos e seções é um pouco diferenciado, pois possui o objetivo apenas de identificar melhor a natureza do parágrafo. A saída gerada auxiliará no entendimento do fonte, facilitando a classificação das regras de negócio.

Maiores detalhes sobre o meta-modelo construído para implementar esta fase do processo podem ser encontrados no Anexo C – Esquema das padronizações para identificação de termos de negócio.

3.3.4 Fase 3 – Extração das regras de BD

O passo seguinte é a extração das regras de BD. Nesta fase ocorre a obtenção de termos de negócio e de todas as regras relacionadas aos bancos de dados. Uma boa parte das heurísticas começa a ser acionada neste momento.

Após inseridos os arquivos oriundos dos fontes e cadastrados os parâmetros para refinar o processo de extração, tudo está preparado para que os primeiros termos e regras sejam obtidas a partir dos bancos de dados. Para os termos especificamente serão aplicadas apenas duas heurísticas, conforme descrito no Quadro 3-3 abaixo.

Heurística RN	Descrição
HRNT01	Cada DATASET gera um termo de negócio.
HRNT02	Cada variável global de BD ou campo de DATASET (seja item de grupo ou não) gera um termo de negócio.

Quadro 3-3 - HRN para identificação e extração de termos de negócio definidos a partir do banco de dados DMSII**.**

Estas heurísticas atuam sobre o resultado da interpretação do arquivo DASDL, onde são descritos os itens de dado persistidos no negócio, ou seja, elementos do banco de dados. Neste caso, apenas datasets, variáveis globais e campos de DATASET são considerados.

Recapitulando, podemos obter algumas regras a partir do banco de dados. São elas fatos, restrições e derivações. As heurísticas para estas regras estão descritas no Quadro 3-4.

Heurística RN	Tipo de RN	Descrição
HRNF01	Fato	Os relacionamentos entre os datasets de programa e seus campos geram fatos do tipo ASSOCIAÇÃO ESTRUT-ITEM.
HRNF02	Fato	Os relacionamentos entre termos que sejam itens de grupo e seus itens elementares geram fatos do tipo AGREGAÇÃO.
HRNF03	Fato	Dois DATASETs distintos que possuam campos que se referenciam ao mesmo termo de negócio geram um fato do tipo ASSOCIAÇÃO ESTRUT-DS.
HRNF06	Fato	Campos de SET ou SUBSET com cláusula DUPLICATES indicam que o DATASET/estrutura/termo de negócio é ordenado pelos nomes dos atributos que compuserem a cláusula KEY, gerando assim um fato do tipo ORDENAÇÃO.
HRNF07	Fato	Estruturas definidas pelo tipo RECORD geram um fato do tipo HERANÇA.
HRNF08	Fato	Se um DATASET possui um SET/SUBSET do tipo NODUP e este SET/SUBSET é parte de outro SET/SUBSET em outro DATASET, então existe um fato do tipo DEPENDÊNCIA este e aquele DATASET.
HRNR01	Restrição	Expressões lógicas existentes na cláusula WHERE de um SUBSET no DASDL estabelecem restrições de acesso.
HRNR02	Restrição	Termos de negócio que apresentem a opção REQUIRED determinam uma restrição de atualização.
HRNR03	Restrição	Expressões lógicas existentes na cláusula VERIFY de um DATASET no DASDL estabelecem restrições de atualização.
HRNR04	Restrição	A cláusula NO DUPLICATES quando ocorre em SETs ou SUBSETs determina uma restrição de atualização.
HRND01	Derivação	Termos derivados de campos de DATASETs de bancos de dados que possuam inicialização diferente do valor *default* geram uma derivação BD do tipo iniciação .
HRND02	Derivação	Itens de dado que sejam iniciados na DATA DIVISION em um programa e apenas referenciados são considerados constantes para o programa.
HRND03	Derivação	Comandos gerados pela cláusula AGGREGATE são considerados como derivações de BD do tipo cálculo.

Quadro 3-4 - Descrição da HRN derivadas de bancos de dados.

Após a extração das regras de BD, o especialista de domínio pode intervir novamente através do procedimento de equivalência de termos ou simplesmente editando as regras para incrementar sua descrição. A equivalência consiste da vinculação de termos obtidos da extração com o mesmo significado. Exemplificando, se forem obtidos dois termos denominados QUANT e QUANTI, o especialista pode identificar que os dois se referem a um único termo denominado QUANTIDADE. Desta forma, todas as regras anteriormente vinculadas a QUANT e QUANTI passarão a se referenciar a QUANTIDADE. Este recurso é importante para minimizar a redundância em relação ao extrator.

3.3.5 Fase 4 – Extração das regras de programa.

A última fase do processo consiste da extração das regras de programa. Nesta etapa, são obtidos termos de negócio que não constavam do modelo de dados (arquivos, tabelas, variáveis). Além deste resultado, conseguimos determinar a abrangência dos termos de negócio (local, programa, sistema ou global) em relação às consultas e atualizações a que é submetido.

Uma vez trabalhado o banco de dados, cada programa deve ser tratado de tal modo a identificarmos termos e regras que estejam referenciados nos fontes COBOL. Para obtenção dos termos a avaliação é realizada sobre a DATA-BASE SECTION, enquanto as regras do modelo dinâmico encontram-se na PROCEDURE DIVISION.

Para identificação dos termos utilizados em um programa, as heurísticas apresentadas no Quadro 3-5 devem ser aplicadas.

Heurística RN	Descrição
HRNT01	Cada ARQUIVO gera um termo de negócio.
HRNT02	Cada campo de ARQUIVO (seja item de grupo ou não) gera um termo de negócio.
HRNT03	Itens de dado da WS-SECTION de programas geram termos de negócio quando identificados como tabelas ou itens com nível 88.
HRNT04	As variáveis globais de um BD declarado em um programa são consideradas termos do programa.
HRNT05	Todos os DATASETS e ARQUIVOS declarados no programa são considerados termos do programa.
HRNT06	Todos os campos de arquivo declarados no programa são considerados termos do programa.
HRNT07	Quando houver declaração de SETs e/ou SUBSETs no programa, todos os campos declarados nos SETs e/ou SUBSETs serão considerados como termos do programa.
HRNT08	Todos os campos referenciados como identificadores de expressão na cláusula WHERE de um SUBSET declarados no programa ou na cláusula VERIFY de um DATASET declarado no programa serão considerados termos do programa.

Quadro 3-5 - HRN obtidas a partir do tratamento das declarações da DATA-BASE SECTION**.**

Sobre as heurísticas acima, cabem algumas informações para esclarecimento:

- Em relação às variáveis internas declaradas no programa, inicialmente, serão consideradas termos de negócio apenas as declaradas em tabelas e as que apresentarem especificação de domínio (nível 88). No primeiro caso, constatamos que nos legados em COBOL, as tabelas muitas vezes são utilizadas como verdadeiros arquivos, sendo carregadas pelo próprio programa em tempo de execução. Ou seja, se não houvesse as restrições de configuração, em um modelo relacional, elas provavelmente

configurariam uma entidade. Já no caso de nível 88, ocorre na descrição do item de dado a especificação sobre seu domínio (valores possíveis), o que eleva a chance de estarmos diante de um termo de negócio;

- Em relação aos DATASETs declarados no programa, estaremos considerando como termos utilizados pelo programa apenas os que efetivamente tiverem sido referenciados no código. Estes termos serão identificados pelas heurísticas de derivação, conforme veremos posteriormente;

- Os campos das estruturas de indexação (SETs e SUBSETs), assim como as expressões lógicas a elas associadas são consideradas termos utilizados no programa por representarem importante informação sobre a ação do programa no banco de dados referenciado.

As heurísticas para regras de programa estão reunidas no Quadro 3-6.

Heurística RN	Descrição
HRNF01	Os relacionamentos entre os arquivos de programa e seus campos geram fatos do tipo ASSOCIAÇÃO ESTRUT-ITEM.
HRNF02	Os relacionamentos entre termos que sejam itens de grupo e seus itens elementares geram fatos do tipo AGREGAÇÃO.
HRNF04	Um DATASET e um arquivo de programa que possuam campos que se referenciam ao mesmo termo de negócio geram um fato do tipo ASSOCIAÇÃO ESTRUT-DS-ARQ.
HRNF05	DATASETs referenciados em um mesmo programa geram um fato do tipo ASSOCIAÇÃO POR PROGRAMA.
HRNR05	Expressões lógicas do comando IF em parágrafos de leitura estabelecem restrições de acesso. São considerados parágrafos de leitura aqueles assim determinados pelo especialista de domínio e os que possuírem comandos do tipo LB e LC.
HRNR06	Expressões lógicas do comando IF em parágrafos de gravação estabelecem restrições de atualização. São considerados parágrafos de leitura aqueles assim determinados pelo especialista de domínio e os que possuírem comandos do tipo AB e AC.
HRNR07	Expressões lógicas existentes na cláusula AT do comando FIND determinam uma restrição de acesso.
HRNR08	Expressões lógicas existentes na cláusula AT dos comandos LOCK e MODIFY determinam uma restrição de atualização.
HRNR09	Expressões lógicas do comando IF em parágrafos de erro ou que habilitem comandos de tratamento de exceção estabelecem restrições de erro.
HRND02	Itens de dado que sejam iniciados na DATA DIVISION em um programa e apenas referenciados são considerados constantes para o programa.
HRND04	Comandos MOVE utilizando as palavras-chave SPACES, HIGHVALUES, LOWVALUES, ZEROS, etc., são caracterizados como derivação de programa do tipo iniciação;
HRND05	Itens de dado que ocorram apenas uma vez no código como uma derivação de iniciação são caracterizados como constantes.
HRND06	Comandos MOVE que possuam valores em sua declaração caracterizem derivações de programa do tipo mudança de estado no item de dado referenciado.
HRND07	Comandos MOVE envolvendo exclusivamente itens de dado cujo(s) domínio(s) seja(m) diferente(s) do atributo consultado caracterizam uma derivação de programa do tipo mudança de domínio.
HRND08	Todos os comandos aritméticos classificados como AA (atualização de atributo) resultam derivações de programa do tipo cálculo.
HRND09	Derivações do mesmo tipo em seqüência ao longo do código de um programa são consideradas como uma única derivação.
HRND10	A primeira derivação simples envolvendo um termo e uma variável fará com que esta se torne um campo associado ao termo e portanto com regras identificáveis.
HRNH01	Comandos do tipo atualização de arquivo dentro de rotinas classificadas como tratamento de erro são classificados como ações de tratamento de erro.
HRNH02	Ações determinadas após a expressão DMSTATUS(NOTFOUND) num comando de acesso ou atualização de banco de dados são classificadas como ações de tratamento de erro.
HRNH03	Expressões lógicas do comando IF que não se enquadrem nas heurísticas HRNR05 (parágrafo de leitura) e HRNR06 (parágrafo de gravação) são classificadas como regras habilitadoras de ação.
HRNH04	Expressões lógicas da cláusula UNTIL do comando PERFORM determinam regras habilitadoras de ação.
HRNH05	Valores inteiros ou itens de dado existentes na cláusula TIMES do comando PERFORM determinam o número de vezes que as ações habilitadas irão executar.

Quadro 3-6 - HRN obtidas através do tratamento dos programas COBOL.

Neste ponto vale ressaltar a importância dos padrões e das naturezas informados pelo especialista de domínio para o tratamento das restrições, a saber:

- São considerados parágrafos de leitura aqueles cuja nomenclatura se enquadre nos padrões definidos pelo especialista ou que possua ocorrência de um comando de leitura de arquivo ou consulta de BD;

- São considerados parágrafos de gravação aqueles cuja nomenclatura se enquadre nos padrões definidos pelo especialista ou que possuam ocorrência de um comando de gravação de arquivo ou atualização de BD;

- São considerados parágrafos de abertura de arquivos aqueles cuja nomenclatura se enquadre nos padrões definidos pelo especialista ou que possuam ocorrência de um comando de abertura de arquivos; e

- São considerados parágrafos de tratamento de erros aqueles cuja nomenclatura se enquadre nos padrões definidos pelo especialista. Consideraremos também sentenças de tratamento de erro aquelas onde expressões lógicas habilitem comandos para o tratamento de exceção.

Seguindo esta etapa do processo, após o tratamento de cada programa, a análise da abrangência dos termos de negócio produzirá um resultado muito interessante para a compreensão do código legado, uma vez que identificando os pontos de leitura e gravação dos termos, estaremos fornecendo uma idéia de matriz CRUD (Create, Read, Update e Delete), conhecimento vital para qualquer projeto de banco de dados que venha a ser feito a partir do sistema legado. Esta fase é dividida em três etapas, cada uma das quais utilizando um conjunto de heurísticas.

As heurísticas referenciadas no Quadro 3-7 procuram identificar os termos a partir dos tipos de comando ou de parágrafo determinando sua abrangência durante a análise dos programas.

Heurística RN	Descrição
HRNT09	Termos referenciados em comandos do tipo leitura de arquivo ou de banco de dados são considerados como termos acessados pelo programa.
HRNT10	Termos referenciados em comandos do tipo atualização de arquivo ou de banco de dados são considerados como termos atualizados pelo programa.
HRNT11	Comandos do tipo atualização têm precedência sobre comandos do tipo leitura na identificação do termo de negócio.[3]
HRNT12	Itens de dados referenciados apenas em um parágrafo/seção de um programa são considerados como acessados localmente.
HRNT13	Itens de dados atualizados apenas em um parágrafo/seção de um programa são considerados como atualizados localmente.
HRNT14	Itens de dados referenciados em mais de um parágrafo/seção de um programa são considerados como acessados pelo programa.
HRNT15	Itens de dados atualizados em mais de um parágrafo/seção de um programa são considerados como atualizados pelo programa.
HRNT16	Itens de dados referenciados em mais de um programa são considerados como acessados pelo sistema.
HRNT17	Itens de dados atualizados em mais de um programa são considerados como atualizados pelo sistema.
HRNT18	Itens de dados referenciados em mais de um sistema são considerados como acessados globalmente.
HRNT19	Itens de dados atualizados em mais de um sistema são considerados como atualizados globalmente.

Quadro 3-7 - HRN para estabelecer abrangência dos itens de dados identificados no código do programa.

[3] Para a definição desta heurística (predominância do WRITE (comandos de gravação) sobre o READ (comandos de leitura) consultamos a taxonomia sobre transações, verificando as características desejáveis numa transação (ACID – Atomicidade, Consistência, Isolamento, Durabilidade) [ÖZSU99]

Finalmente a abrangência dos agrupamentos de dados (DATASETs e arquivos do programa), pode ser determinada aplicando-se as heurísticas apresentadas no Quadro 3-8.

Heurística RN	**Descrição**
HRNT20	Arquivos ou DATASETs que possuam campos atualizados apenas por um programa são considerados arquivos/DATASETs atualizados pelo programa.
HRNT21	Arquivos ou DATASETs que possuam pelo menos um campo atualizado por programas diferentes de um mesmo sistema são considerados arquivos/DATASETs atualizados pelo sistema.
HRNT22	Arquivos ou DATASETs que sejam atualizados por mais de um sistema são considerados arquivos/DATASETs atualizados globalmente
HRNT23	Arquivos ou DATASETs que possuam campos acessados apenas por um programa são considerados arquivos/DATASETs acessados pelo programa.
HRNT24	Arquivos ou DATASETs que possuam apenas campos acessados por programas de um sistema são considerados arquivos/DATASETs acessados pelo sistema.
HRNT25	Arquivos ou DATASETs que possuam apenas campos acessados por mais de um sistema são considerados arquivos/DATASETs acessados globalmente.
HRNT26	Termos do Programa são compostos por campos dos arquivos de um programa e datasets referenciados apenas no programa.
HRNT27	Termos de Sistema são compostos por campos de arquivos de sistema, bancos de dados e DATASETs utilizados por mais de um programa no mesmo sistema.
HRNT28	Termos Globais são compostos por campos de arquivos globais, bancos de dados (e suas variáveis globais) utilizados por mais de um sistema e DATASETs utilizados em mais de um sistema

Quadro 3-8 - HRN para caracterizar os arquivos e datasets do código legado.

Continuando fiel ao processo semi-automático de identificação de regras, o especialista pode novamente alterar a descrição das regras obtidas, excluir as que não forem relevantes e agregar valor ao repositório adicionando seu conhecimento sobre o negócio e sobre o sistema legado.

Conhecidas as heurísticas, o processo e um pouco do ambiente UNISYS, temos subsídios suficientes para descrever a ferramenta **Source Inspector**.

3.4 Source Inspector

A heterogeneidade dos sistemas legados provoca grande dificuldade para se eleger uma única solução mais abrangente em relação à abordagem a ser utilizada na identificação de componentes e objetos de negócio [VERH00].

Portanto, em vez de estabelecer uma mistura das soluções existentes, optamos por *focar* o problema fornecendo uma proposta de abordagem mais centrada em nosso problema específico. Para o escopo deste estudo, o maior interesse é a tentativa de estabelecer uma solução menos abrangente e mais customizada para o ambiente alvo (UNISYS/COBOL/DMSII). Neste contexto, a solução escolhida foi o desenvolvimento do **Source Inspector**, um software específico para leitura de fontes gerados no ambiente UNISYS [GAVI00].

Esta ferramenta faz análise de sistemas legados que utilizem linguagem COBOL, acessando ou não bases de dados DMSII. A partir dos fontes obtidos na entrada, o **Source Inspector** fornece uma proposta de modelo das classes de domínio do sistema, uma relação das anomalias identificadas durante a passagem para o paradigma OO, um conjunto de medidas estatísticas do sistema analisado e um conjunto de termos e regras de negócio devidamente agrupadas e extraídas a partir de heurísticas [SIQU01].

Internamente, a ferramenta é composta por dois módulos: o Módulo Tradutor (ModTra) e o Módulo Extrator (ModEx), conforme pode ser observado na Figura 4.1.

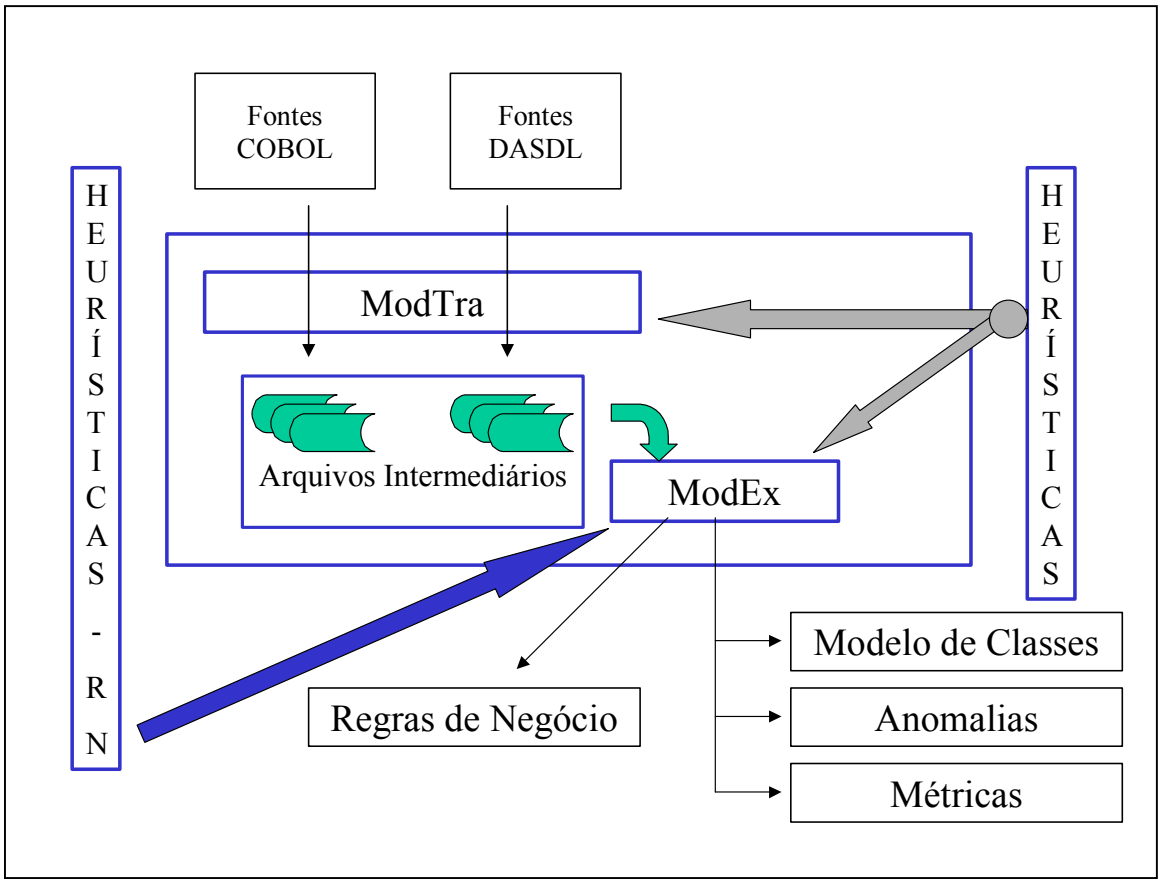

Figura 3.8 - **Estrutura interna do Source Inspector (Fonte: [SIQU02])**

3.4.1 O Módulo Tradutor (ModTra)

O ModTra foi desenvolvido em linguagem C a partir do analisador léxico LEX e do analisador sintático YACC, ambos utilizados no ambiente Unix. [AHO86, KERN86, LEVI95, SCHI96]. Em termos de código, trata-se apenas de quatro programas, mas que compõem 12.000 linhas de código.

A partir do arquivo DASDL, o ModTra extrai informações dos bancos de dados, variáveis globais e datasets (assim como seus campos, sets e

subsets). Dos fontes COBOL, são extraídos os arquivos usados pelo programa, variáveis locais, seções, parágrafos, comandos e campos por eles acessados. Se o programa COBOL acessar algum banco de dados, também são identificadas as estruturas DMSII referenciadas (datasets, sets e subsets).

Para o correto funcionamento do ModTra, três premissas são necessárias:

- Código corretamente compilado (sem erros). Os objetos submetidos devem estar sem quaisquer erros de compilação. Assume-se que os fontes já estejam corretamente compilados e, de preferência, em produção;
- Código fonte digitado em caixa alta (caracteres maiúsculos), excetuando os literais (seqüências de caracteres entre aspas) quando for o caso;
- Fazer primeiro o tratamento dos arquivos DASDL e apenas depois o dos arquivos COBOL;

Ao final, o ModTra gera como saída um arquivo intermediário no formato texto para cada fonte (DASDL ou COBOL) recebido como entrada. Para fontes DASDL, o arquivo intermediário apresenta os seguintes tipos de registro:

Tipo Reg	**Nome**	**Descrição**
D0	BD	Nome de banco de dados
D1	DATASET	Nome do DATASET
D2	CAMPO	Detalhes sobre os campos do DATASET
D3	CHAVE	Nome do SET e do DATASET a que pertence
D4	CAMPOS_SET	Nome dos campos do SET

D5	ESTATIST1	Estatísticas de totalização [4]
D6	ESTATIST2	Estatísticas sobre utilização das heurísticas para extração do modelo de classes
D7	GLOBALBD	Nome dos itens de dados globais
D8	VARGLOBAL	Detalhes sobre os itens de dados globais
D9	SUBSET	Detalhes sobre o SUBSET
D10	EXPRESSAO	Expressão relacionada ao SUBSET
D11	CAMPOS_SUBSET	Campos relacionados no SUBSET
D12	IDENT_EXPR	Identificadores da expressão
D13	CAMPO_DATA_SET	Campos referenciados na cláusula DATA do SET
D14	CAMPO_DATA_SUBSET	Idem para o SUBSET
D15	COMENTARIO	Estrutura existente para conter linhas de comentários dos itens de dado de um DASDL
D16	DEFAULTS	Valores default para os diferentes formatos de campo em um BD
D17	AGGREGATE	Cláusulas de totalização

Quadro 3.9 **– Tipos de registro gerados pelo ModTra a partir dos arquivos DASDL.**

Para fontes COBOL, os tipos de registro são os seguintes:

Tipo Reg	**Nome**	**Descrição**
C0	PROGRAMA	Nome do programa
C1	ARQUIVO	Nome e identificador de arquivo de programa
C2	CAMPO	Detalhes sobre os campos dos arquivos de programa

[4] Número de DATASETs, número de campos (mínimo, máximo e médio por DATASET), totais por tipo de campo (alfabético, inteiro, decimal, booleano, grupo) e número de sets (mínimo, máximo e médio por DATASET);

C3	BD	Nome de banco de dados do programa
C4	DATASET	Nome de DATASET do programa
C5	INDICE	Nome de SET ou SUBSET de programa
C6	VARIAVEL	Detalhes sobre as variáveis do programa
C7	SENTENCA	Detalhes sobre as sentenças do programa
C8	CAMPO_COMANDO	Detalhes sobre os campos de cada comando COBOL
C9	SECAO	Nomes de seções
C10	PARAGRAFO	Nomes dos parágrafos da PROCEDURE DIVISION
C11	ESTATISTICAS I	Estatísticas das variáveis do programa
C12	ESTATISTICAS II	Estatísticas das variáveis de BD utilizadas
C13	ESTATISTICAS III	Estatísticas sobre as linhas de comentário
C14	HEURISTICAS	Uso das heurísticas para identificação de classes
C15	COMANDO_COND	Detalhes sobre os comandos condicionais
C16	EXPRESSAO	Linhas das expressões identificadas
C17	IDENT_EXPR	Seqüência de identificadores das expressões

Quadro 3.10 – **Tipos de registro gerados pelo ModTra a partir dos arquivos COBOL.**

3.4.2 O Módulo Extrator (ModEx)

Este módulo foi implementado em 28.550 linhas de código Delphi e utiliza o banco de dados Paradox [CANT98, TEIX00]. Ele lê o arquivo fornecido pelo ModTra e a partir de suas informações gera a proposta para o modelo de classes, determina as anomalias existentes, registra as medidas do sistema e as RN. O processo de carga é executado em duas fases. Na primeira fase, as informações contidas nos arquivos intermediários são armazenadas em um banco de dados. Na segunda fase, aplicam-se as heurísticas para tratamento das informações dos bancos de dados e dos programas COBOL.

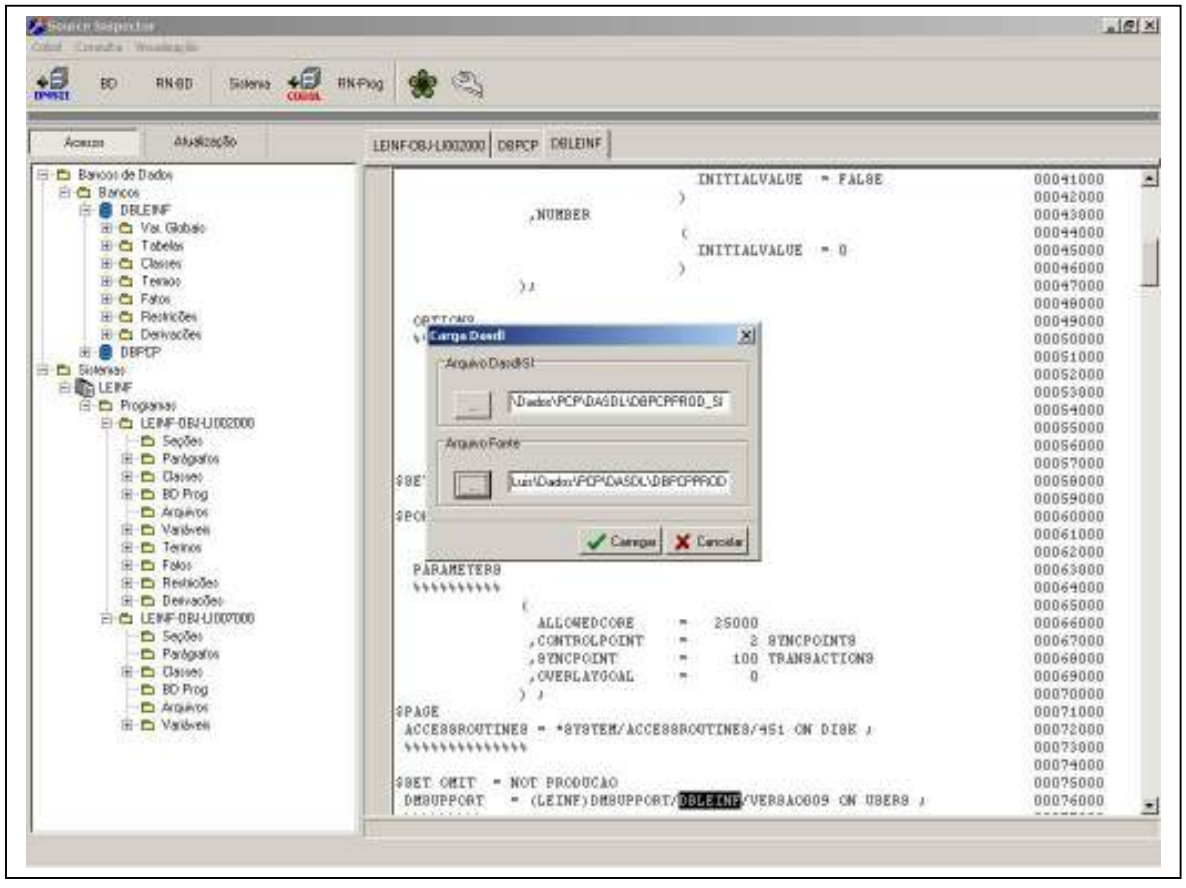

Figura 3.9 – **Parametrização para carga inicial de um fonte no** Source Inspector**. O usuário informa o caminho para o arquivo intermediário e o fonte correspondente.**

As heurísticas de [GAVI00] estão fortemente direcionadas para obtenção de atributos das classes. Este caminho é compreensível porque o modelo de dados é notadamente mais estável que o modelo de processos de um sistema. As alterações em função da mudança de requisitos do negócio ocorrem muito mais freqüentemente nos processos e, conseqüentemente, nos programas em um sistema legado, do que nos bancos de dados propriamente ditos. As alterações envolvendo mudanças de atributos são inclusive consideradas mais críticas, tendo um acompanhamento muito mais apurado por parte dos desenvolvedores. Das 26 heurísticas definidas por [GAVI00], 12 por serem aplicadas diretamente sobre o DMSII fazem referência a atributos. Das 14 restantes, apenas quatro tratam da extração dos métodos, mesmo assim de forma rudimentar, apenas concatenando sentenças de programa.[SIQU01]

4 RESULTADOS DO ESTUDO DE CASO COM O SOURCE INSPECTOR

"... o desenvolvimento é uma invenção
própria de seu destino,
não é a reprodução
do que já se conhece."

Celso Furtado
Economista

"O pessimista queixa-se do vento.
O otimista espera que ele mude.
O realista ajusta as velas."

Willian George Ward (1812-1992)
Teólogo inglês

4.1 Validação da ferramenta

4.1.1 Obtenção de fontes e bancos de dados

Nesta etapa fizemos a avaliação de quais fontes seriam interessantes para testar o Módulo Tradutor da ferramenta. Inicialmente foram escolhidos três sistemas do BC, todos utilizados na sala de operações (SALOP) do Departamento de Operações do Mercado Aberto. As características dos sistemas são descritas a seguir:

- SIG – Sistema de Informações Gerenciais

O SIG é um sistema eminentemente *on-line* onde os operadores da SALOP monitoram oscilações de alguns indicadores do mercado.

- OFPUB – Oferta Pública Formal Eletrônica

O OFPUB é um dos sistemas mais estratégicos da área de negócio de operações do mercado aberto. Ele é o responsável pela colocação primária e secundária de títulos públicos do governo federal, possibilitando assim a captação de recursos para custeio e financiamento dos compromissos da União.

- LEINF – Leilão Informal

O LEINF é um sistema de leilão utilizado pelo BC junto às instituições financeiras credenciadas (*dealers*) para a execução da política monetária controlando o nível de liquidez da economia.

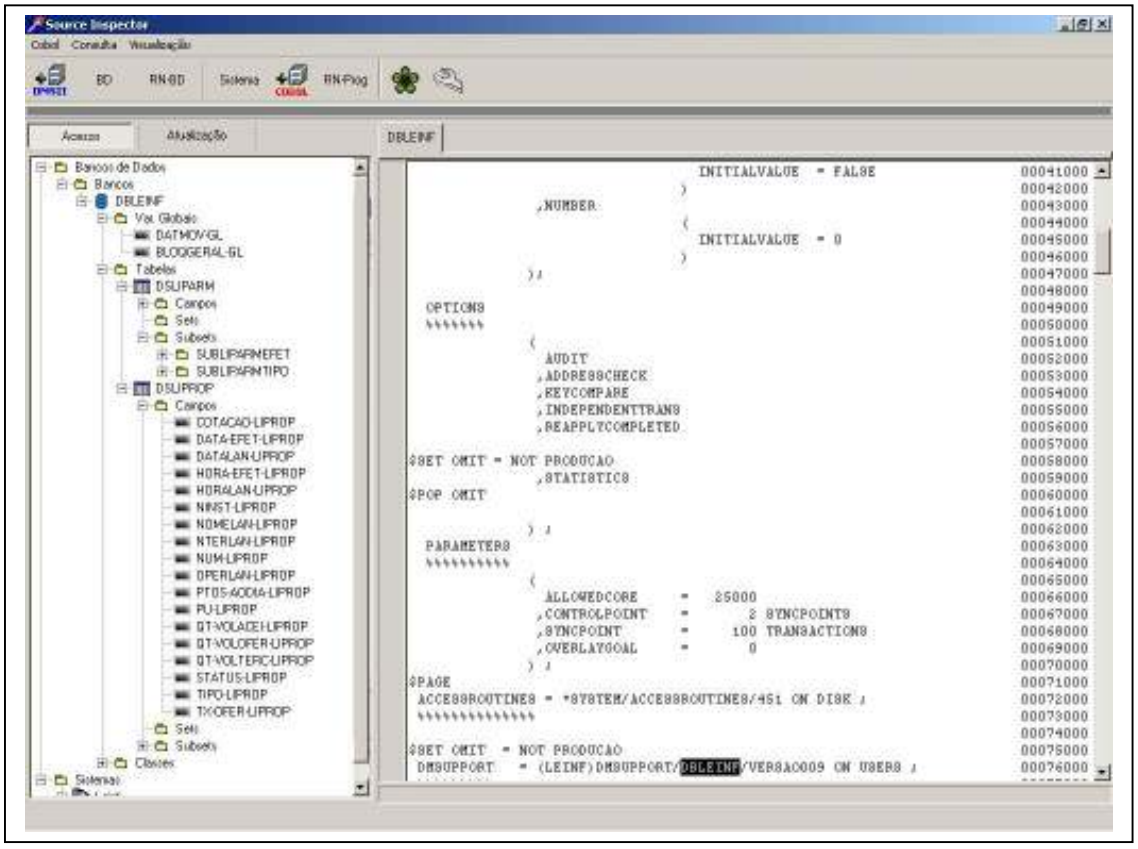

Figura 4.1 – **Visualização do** Source Inspector **após a carga do DBLEINF, suas variáveis globais, seus dois datasets (DSLIPARM e DSPROP) e os campos destes.**

Estes sistemas não possuem o porte nem a complexidade do **SELIC**, mas contêm regras de negócio críticas para o BC. Eles foram escolhidos por terem os especialistas de domínio mais disponíveis para validação das regras obtidas, além de se enquadrarem no perfil de sistema legado descrito no capítulo 2. A tabela abaixo mostra os primeiros resultados obtidos a partir da execução do Módulo Tradutor.

Sistema	Banco de Dados	Número de DATASETs	Número de Sets	Número de Subsets	Número de Campos
SIG	DBSIG	10	10	0	258
LEINF	DBLEINF	2	0	9	67
OFPUB	DBOFPUB	9	7	12	164

Quadro 4.1 – **Resultado da execução do Módulo Tradutor DASDL sobre os fontes BC.**

Sistema	Número de Programas	Número de Bibliotecas	Num Linhas Maior Programa	Num Linhas Menor Programa	Tamanho Médio dos Programas
SIG	7	11	6066	370	2795
LEINF	14	15	3852	544	1615
OFPUB	33	15	2622	52	802

Quadro 4.2 – **Resultado da execução do Módulo Tradutor COBOL sobre os fontes BC.**

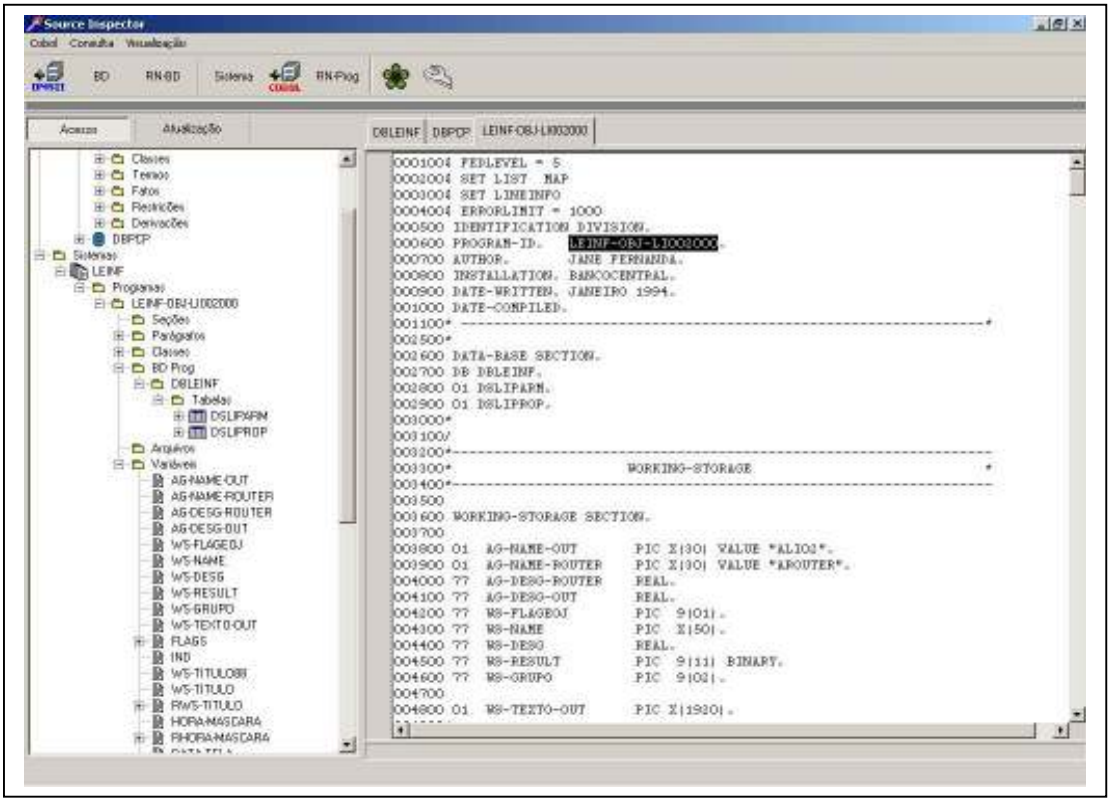

Figura 4.2 – **Visualização do** Source Inspector **após a carga do programa LEINF-OBJ-LI002000, pertencente ao sistema LEINF.**

Os sistemas da SALOP já representariam importante campo de trabalho para obter os resultados desejados para o BC. Entretanto, procurando uma validação mais completa da ferramenta, aplicamos o Módulo Tradutor do **Source Inspector** a fontes de outras instalações com ambiente similar. A mais significativa delas é a DATAPREV (Empresa de Processamento de Dados do Ministério da Previdência Social). A instalação UNISYS da DATAPREV é a maior do Brasil. Seus volumes de informação são muitas vezes maiores que os processados pelos sistemas da SALOP. Seus sistemas legados também estão dentre os mais complexos para a plataforma COBOL/DMSII. Portanto, com o objetivo de submeter a ferramenta a um teste de robustez mais acurado, selecionamos alguns bancos de dados da DATAPREV. Estes fontes DASDL são extensos, complexos e alguns deles são gerados por ferramentas automáticas de análise de dados. Neste contexto, o **Source Inspector** gerou as seguintes informações, conforme consta do Quadro 4.5:

Banco de Dados	**Número de DATASETs**	**Número de Sets**	**Número de Subsets**	**Número de Campos**
BDAGREGEMPR	24	26	4	615
BDBENCRED	28	51	5	797
BDCNTCIN	31	60	3	733
BDSISBEN	188	296	84	4559
BDTAB	53	127	46	740
BDTRAB	31	51	13	702
BDVINC	67	124	0	1541
BDVINCCEF	12	20	0	383

Quadro 4.3 **– Resultado da execução do Módulo Tradutor DASDL sobre os fontes DATAPREV.**

Uma vez que o Módulo Tradutor suportou bem o volume de dados dos sistemas da DATAPREV, para efeito de continuação acadêmica dos trabalhos, obtivemos também os fontes do sistema de controle de pessoal do Núcleo de Computação Eletrônica da UFRJ. Estes sistema foi utilizado como massa de dados em [GAVI00]. O quadro abaixo mostra o resultado gerado pelo Módulo Tradutor para o banco de dados deste sistema:

Sistema	**Banco de Dados**	**Número de DATASETs**	**Número de Sets**	**Número de Subsets**	**Número de Campos**
PCP	DBPCP	39	47	8	350

Quadro 4.4 – **Resultado da execução do Módulo Tradutor DASDL sobre o fonte do sistema PCP.**

ROSA – Como conseguir este resultado

O primeiro grande resultado, portanto, foi tornar o **Source Inspector** um aplicativo robusto para fontes de quaisquer instalações UNISYS. Este é um resultado bastante importante porque a premissa do produto é a de que os fontes não devem ser alterados para se adequar à ferramenta. Uma vez estando no ambiente UNISYS, basta que sejam gerados arquivos texto correspondentes que o Módulo Tradutor, no ambiente Unix, produzirá os arquivos intermediários necessários para processamento no Módulo Extrator.

4.1.2 Implementação das modificações no Módulo Tradutor

ROSA – necessidade ou resultado achado / avaliação (análise dos resultados / interfere ... dos dados

Depois que o Módulo Tradutor do **Source Inspector** foi transformado num produto mais robusto tornou-se necessário implementar uma série de regras gramaticais para a geração de informações que pudessem revelar regras de negócio.

Estas novas regras gramaticais foram introduzidas tanto no aplicativo que trata dos fontes DASDL (ModTra-DASDL), quanto no aplicativo que trata dos fontes COBOL (ModTra-COBOL).

- Novas regras gramaticais para o ModTra-DASDL

  - Tipo de registro DEFAULTS

    Os bancos de dados DMSII permitem que o valor para seus campos seja pré-definido de acordo com o formato. Se esta cláusula for declarada no banco de dados, podem ser definidos valores default para o tipo ALPHA, NUMBER, etc. Este tratamento permite ao **Source Inspector** definir regras de iniciação para os itens de dados, o que facilita a compreensão do código.

  - Tipo de registro AGGREGATE

    A cláusula AGGREGATE não é muito utilizada pelos exemplos com que trabalhamos, mas constitui-se em uma das poucas *triggers* do banco DMSII. Através desta cláusula são gerados totalizadores envolvendo itens de dados de um determinado DATASET. Temos assim um exemplo de regra de derivação que conceitualmente deve ser contemplado no **Source Inspector.**

  - Tipo de registro COMENTARIO

    Descobrimos a importância dos comentários no DASDL ao analisarmos os fontes gerados para os bancos de dados da

DATAPREV. Os comentários no DASDL podem ser feitos aleatoriamente (utilizando o caracter '%') ou especificamente após alguns itens do DASDL, tais como datasets, sets, subsets e campos de DATASET. Este recurso permite ao **Source Inspector** definir melhor os termos de negócio obtidos no banco de dados porque pode obter informações diretamente escritas pelo programador ou especificador do programa.

- Tipo de registro SUBSET

A primeira versão do **Source Inspector** simplesmente não tratava esta estrutura. É importante observar que o SUBSET restringe o escopo de um programa, sendo importantíssimo para extração de regras de negócio.

- Tratamento de Expressões

As expressões lógicas são parte importante das regras de negócio do tipo restrição. No DASDL elas são obtidas em dois pontos específicos: na definição de datasets e na cláusula WHERE de subsets. No primeiro caso, um DATASET (ou quaisquer de seus tipos de registro) pode apresentar a cláusula VERIFY, que determinará uma regra específica atrelada a uma expressão condicional para que o DATASET seja atualizado. Trata-se neste caso de uma restrição de atualização. No segundo caso, temos a expressão lógica que estabelece a seleção horizontal citada no conceito de SUBSET no capítulo 3. Trata-se portanto de uma restrição de acesso.

- Cláusula DUPLICATES

Esta cláusula pode aparecer tanto na definição de sets quanto de subsets. Ela determina se um SET ou SUBSET permite chaves duplicadas. Desta forma podemos determinar a ordenação e a identificação de datasets, estabelecendo assim regras de negócio do tipo *fato*.

- Cláusula ORDEM

Definida para campos de SET, esta cláusula indica se existe ordenação para o SET especificado e, em caso afirmativo, se a ordem é ascendente ou descendente. Junto com a cláusula DUPLICATES, auxilia a descrição de fatos como regras de negócio.

- Tratamento de valores iniciais, nulos e campos obrigatórios

Estas três informações foram definidas para os campos de um DATASET. No DASDL, elas são combinadas com as informações declaradas no registro DEFAULTS. Para cada campo, é permitido que valores iniciais sejam definidos, que valores determinados sejam associados a valores nulo e que o campo seja declarado como obrigatório (cláusula REQUIRED). Os valores iniciais e nulos determinam apenas regras de inicialização, mas a obrigatoriedade de um campo determina uma restrição de gravação.

- Novas regras gramaticais para o ModTra-COBOL

- Comando Condicional

Comandos condicionais são aqueles que possuem alguma expressão condicional. Basicamente na gramática do COBOL, determinamos os seguintes tipos de comando condicional:

  - IF
  - PERFORM UNTIL
  - FIND, DELETE, LOCK, MODIFY
  - ON EXCEPTION

A condição lógica existente no comando IF pode determinar uma restrição ou uma regra habilitadora de ação, conforme descrição existente no capítulo 4.

A cláusula UNTIL existente no comando PERFORM estabelece uma condição de parada para a execução de determinada ação. Estas informações facilitam a compreensão do programa conforme observaremos a seguir na explicação sobre estrutura de chamadas de cada programa.

Os comandos FIND, DELETE, LOCK e MODIFY fazem parte da interface que o COBOL possui com o DMSII [UNIS96b, UNIS96c]. Através da cláusula AT determinamos uma expressão que resultará em uma restrição de acesso (FIND e LOCK) ou de atualização (DELETE e MODIFY).

Finalmente a cláusula ON EXCEPTION determina um conjunto de ações a serem tomadas caso o acesso ao banco de dados retorne um erro. Esta cláusula se enquadra nas rotinas de tratamento de erros que, muitas vezes apresentam regras de negócio que podem ser selecionadas pelo especialista de domínio.

- Lista de chamadas

Para uma melhor compreensão do programa, geramos em nosso analisador sintático uma lista que registra as chamadas realizadas em cada seção/parágrafo de um programa. Os elementos da lista são acionados através dos comandos PERFORM e GOTO. Cada estrutura desta traz ainda o nível de profundidade (determinado pelo comando IF) para cada rotina do programa. Exemplificando, dado o pedaço de código abaixo extraído da PROCEDURE DIVISION de um programa hipotético:

```
SECAO_1.
      PERFORM PARAGRAFO_A1.
      PERFORM PARAGRAFO_A2.
PARAGRAFO_A1.
      PERFORM PARAGRAFO_A11.
      PERFORM PARAGRAFO_A12.
SECAO_2.
      PERFORM PARAGRAFO_B1.
      PERFORM PARAGRAFO_B2.
```

Teríamos como resultado da análise, uma lista de seções parágrafos da seguinte maneira:

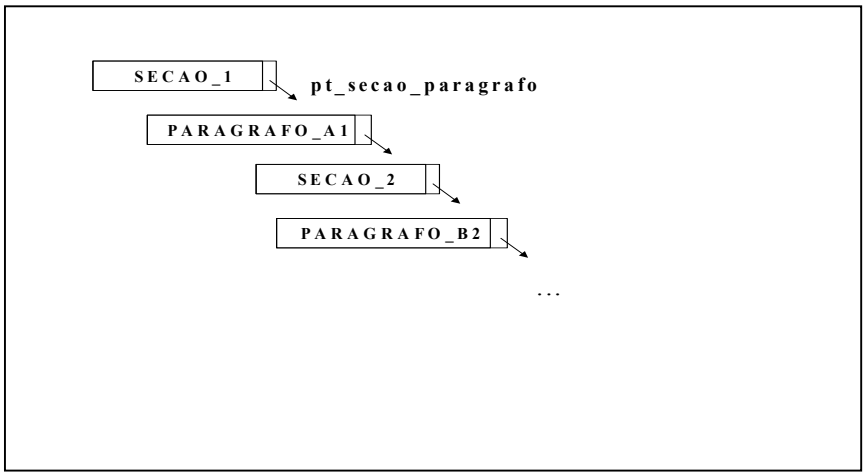

Figura 4.3 – **Seqüência de seções e parágrafos.**

Não estamos considerando aqui onde estão localizados os parágrafos não declarados no exemplo (A2, A11, A12, B21, B22). A estrutura criada produzirá, a partir desta lista, uma matriz com o seguinte formato:

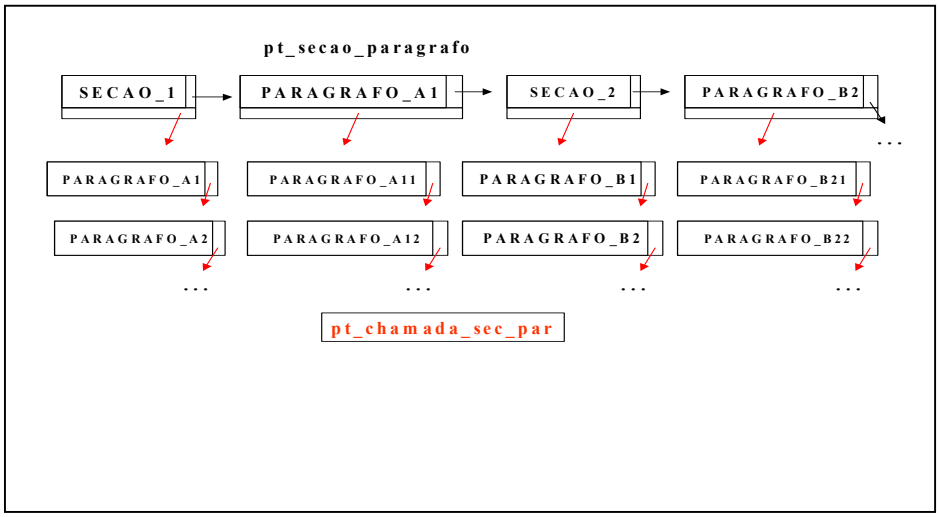

Figura 4.4 – **Árvore gerada pelo ponteiro** pt_chamada_sec_par.

- Profundidade e tipo de regra de negócio

  Estes dois atributos tiveram que ser definidos para permitir uma melhor identificação dos comandos e da profundidade do código. A estrutura alterada é a de *campo-comando*.[GAVI00]

- Sinônimos

Bancos de dados e datasets quando declarados explicitamente em um programa COBOL podem utilizar-se de estruturas similares. Esta é uma alternativa que a linguagem permite para que outros bancos e estruturas físicos sejam utilizados com a mesma definição lógica. Foi necessário alterar o **Source Inspector** para que este pudesse compreender esta estrutura gramatical.

4.1.3 Implementação das heurísticas no Módulo Extrator

Para inserir as heurísticas na versão anterior do **Source Inspector** foi necessário realizar uma grande alteração no modelo lógico da ferramenta. Estas alterações envolveram três atividades básicas:

- Alterações para suportar as novas regras gramaticais;
- Criação de novas entidades para persistir as regras de negócio e para receber os parâmetros definidos pelo especialista de domínio;
- Adaptação da interface para visualização das regras de forma hierárquica.

Resumidamente, conforme mostra o Quadro 5.5 as heurísticas podem ser divididas conforme sua aplicação por categoria de regra de negócio extraída:

Regra de Negócio	**Sigla**	**Número de heurísticas**
Termos	HRNT	28
Fatos	HRNF	8
Restrições	HRNR	9
Derivações	HRND	9
Habilitadoras de Ação	HRNH	5
TOTAL		59

Quadro 4.7 – **Heurísticas por regra de negócio**

Observamos que a maior parte das heurísticas se refere à extração de termos de negócio. Esta constatação é compatível com a proposta de nossa solução que utiliza a abordagem centrada em dados (*Data Centered Approach*) [HUAN98]. Para verificarmos a aplicação destas heurísticas, utilizaremos como exemplo uma parte do sistema OFPUB do BC, mais especificamente a rotina que trata da apuração direta dos leilões de títulos públicos. Este exemplo foi escolhido por conter uma importante rotina de rateio[5].

4.1.4 Demonstração do funcionamento do Source Inspector

O primeiro passo do funcionamento da ferramenta é a extração de termos de negócio a partir do arquivo DASDL[6] que descreve parte do banco **DBOFPUB**. A extração pode ser feita com ou sem a interferência do especialista (cadastrando parâmetros). A extração sem intervenção do especialista gerou os seguintes termos de negócio, conforme o Quadro 4.8.

Em seguida, verificamos que o tratamento da WORKING-STORAGE SECTION (heurística HRNT03) gerou mais alguns termos de negócio quando o programa **ProcessaApuracaoDireta** foi lido pelo **Source Inspector**, conforme pode ser observado no Quadro 4.9. Vale notar que não havia arquivos declarados na FILE SECTION.

---

[5] Maiores detalhes sobre a lógica deste processo podem ser encontrados no Anexo D – Apuração Direta no Sistema OFPUB.

[6] A descrição do banco de dados considerado e o fonte considerado encontram-se no Anexo E – Descrição do Banco de Dados DBOFPUB no formato Dasdl.

Termo de Negócio	Origem	Heurística
DATMOV-GL	Variável Global	HRNT02
PRCDIR-GL	Variável Global	HRNT02
PRCRED-GL	Variável Global	HRNT02
QTPROP-GL	Variável Global	HRNT02
SLIBER-GL	Variável Global	HRNT02
DSPROP	DATASET DSPROP	HRNT01
DSOFER	DATASET DSOFER	HRNT01
CODINS-PROP	Campo DSPROP	HRNT02
COT-PROP	Campo DSPROP	HRNT02
COTPROPOSTO-PROP	Campo DSPROP	HRNT02
DATALAN-PROP	Campo DSPROP	HRNT02
E-PROP	Campo DSPROP	HRNT02
FLAGALT-PROP	Campo DSPROP	HRNT02
FLAGBC-PROP	Campo DSPROP	HRNT02
NUMPRO-PROP	Campo DSPROP	HRNT02
PAPEL-PROP	Campo DSPROP	HRNT02
PU-PROP	Campo DSPROP	HRNT02
PUPROPOSTO-PROP	Campo DSPROP	HRNT02
QUANTA-PROP	Campo DSPROP	HRNT02
QUANTC-PROP	Campo DSPROP	HRNT02
QUANTP-PROP	Campo DSPROP	HRNT02
STATUS-PROP	Campo DSPROP	HRNT02
DATCOM-OFER	Campo DSOFER	HRNT02
DATEMIS-OFER	Campo DSOFER	HRNT02
DATLAN-OFER	Campo DSOFER	HRNT02
DATLIQ-OFER	Campo DSOFER	HRNT02
HIST-OFER	Campo DSOFER	HRNT02
MODULO-OFER	Campo DSOFER	HRNT02
NUMCOM-OFER	Campo DSOFER	HRNT02
PAPEL-OFER	Campo DSOFER	HRNT02
PRCDIR-OFER	Campo DSOFER	HRNT02
SIT-OFER	Campo DSOFER	HRNT02
STATUS-OFER	Campo DSOFER	HRNT02
TIPO-OFER	Campo DSOFER	HRNT02
TITULO-OFER	Campo DSOFER	HRNT02
VENCTO-OFER	Campo DSOFER	HRNT02

Quadro 4.8 – **Termos de negócio extraídos de DBOFPUB.**

Termos de Negócio	
CID-AUX	QTF-F
CORTE-PRE	QTP-AUX
E-AUX	QTP-I
INST-AUX	QTR-F
INST-I	TAB-AUX
NPP-I	TAB-F
NUM-AUX	TABFXA
ORDI-AUX	TAB-I
ORDII-AUX	TAB-II
ORDP-I	TABINS
PRECORTE	TX-AUX
PU-AUX	TX-F
PU-F	VFA-AUX
QI-F	VFA-I
QPD-F	VFC-AUX
QP-F	VFC-I
QTA-AUX	VFD-F
QTA-I	VFF-F
QTC-AUX	VFP-AUX
QTC-I	VFP-I
QTD-F	VFR-F
QTF-F	

Quadro 4.7 – **Termos de negócio extraídos da WS SECTION do programa ProcessaApuracaoDireta, utilizando a heurística HRNT03.**

Um dos recursos interessantes do **Source Inspector** é a possibilidade de cadastramento de alguns parâmetros que facilitam a obtenção dos termos de negócio. Em uma segunda simulação, o especialista inseriu as seguintes regras:

- Desconsiderar para os nomes dos datasets o prefixo DS;
- Desconsiderar para os nomes de campos de DATASET os sufixos após o caracter separador "-";
- Desconsiderar para os itens de tabela os sufixos após o caracter separador "-";
- Considerar as seguintes naturezas para os atributos:
    - COD – Código.
    - COM – Comunicado;

- COT – Cotação;
- DAT – Data;
- DATA – Data;
- DIR – Direta;
- DT – Data;
- LAN – Lançamento;
- LIQ – Liquidação;
- NUM – Número;
- PRC – Preço;
- PU – Preço Unitário;
- QT – Quantidade;
- QUANT – Quantidade;
- RED – Redistribuição;
- SIT – Situação;
- TX – Taxa;
- VF – Valor Financeiro;

Depois de fornecidas estas informações, temos o seguinte resultado:

Termo de Negócio	Campo(s) Origem
CID	CID-AUX
CódigoINS	CODINS-PROP
CORTE-PRE	CORTE-PRE
Cotação	COT-PROP
CotaçãoPROPOSTO	COTPROPOSTO-PROP
DataComunicado	DATCOM-OFER
DataEMIS	DATEMIS-OFER
DataLançamento	DATLAN-OFER, DATALAN-PROP
DataLiquidação	DATLIQ-OFER
DataMOV	DATMOV-GL
E	E-PROP, E-AUX
FLAGALT	FLAGALT-PROP
FLAGBC	FLAGBC-PROP
HIST	HIST-OFER
INST	INST-AUX, INST-I
MODULO	MODULO-OFER
NPP	NPP-I
Número	NUM-AUX
NúmeroComunicado	NUMCOM-OFER
NúmeroPRO	NUMPRO-PROP
OFER	DSOFER
ORDI	ORDI-AUX
ORDII	ORDII-AUX
ORDP	ORDP-I
PAPEL	PAPEL-OFER, PAPEL-PROP
PreçoDireta	PRCDIR-GL, PRCDIR-OFER
PreçoRedistribuição	PRCRED-GL
PRECORTE	PRECORTE
PreçoUnitárioPROPOSTO	PUPROPOSTO-PROP
PROP	DSPROP
PU	PU-PROP, PU-AUX, PU-F
QI	QI-F
QP	QP-F
QPD	QPD-F
QuantidadeA	QUANTA-PROP, QTA-AUX, QTA-I
QuantidadeC	QUANTC-PROP, QTC-AUX
QuantidadeD	QTD-F
QuantidadeF-F	QTF-F
QuantidadeP	QUANTP-PROP, QTP-AUX, QTP-I
QuantidadePROP	QTPROP-GL
QuantidadeR	QTR-F
Situação	SIT-OFER
SLIBER	SLIBER-GL
STATUS	STATUS-OFER, STATUS-PROP
TAB-AUX	TAB-AUX
TAB-F	TAB-F
TABFXA	TABFXA
TAB-I	TAB-I
TAB-II	TAB-II
TABINS	TABINS
Taxa	TX-AUX, TX-F
TIPO	TIPO-OFER
TITULO	TITULO-OFER
ValorFinanceiroA	VFA-AUX, VFA-I
ValorFinanceiroC	VFC-AUX, VFC-I
ValorFinanceiroD	VFD-F
ValorFinanceiroF	VFF-F
ValorFinanceiroP	VFP-AUX, VFP-I
ValorFinanceiroR	VFR-F
VENCTO	VENCTO-OFER

Quadro 4.9 – **Relação final dos termos de negócio após a aplicação dos padrões.**

Mesmo após a inclusão de parâmetros, o resultado da extração de termos de negócio pode ainda ser refinado. Neste momento, o especialista pode novamente intervir para relacionar uma série de campos similares aos mesmos termos de negócio, num procedimento que denominamos *equivalência de termos*. Na verdade a figura do especialista no primeiro momento pode ser representada pelo especialista do negócio em si, aquele que possui o completo domínio sobre os termos do negócio. Neste segundo momento, é mais indicado que a interação com o **Source Inspector** seja realizada pelo analista responsável pelo sistema legado. Estes profissionais geralmente possuem um bom conhecimento do domínio e ainda um razoável entendimento do código e da nomenclatura dos itens de dados.

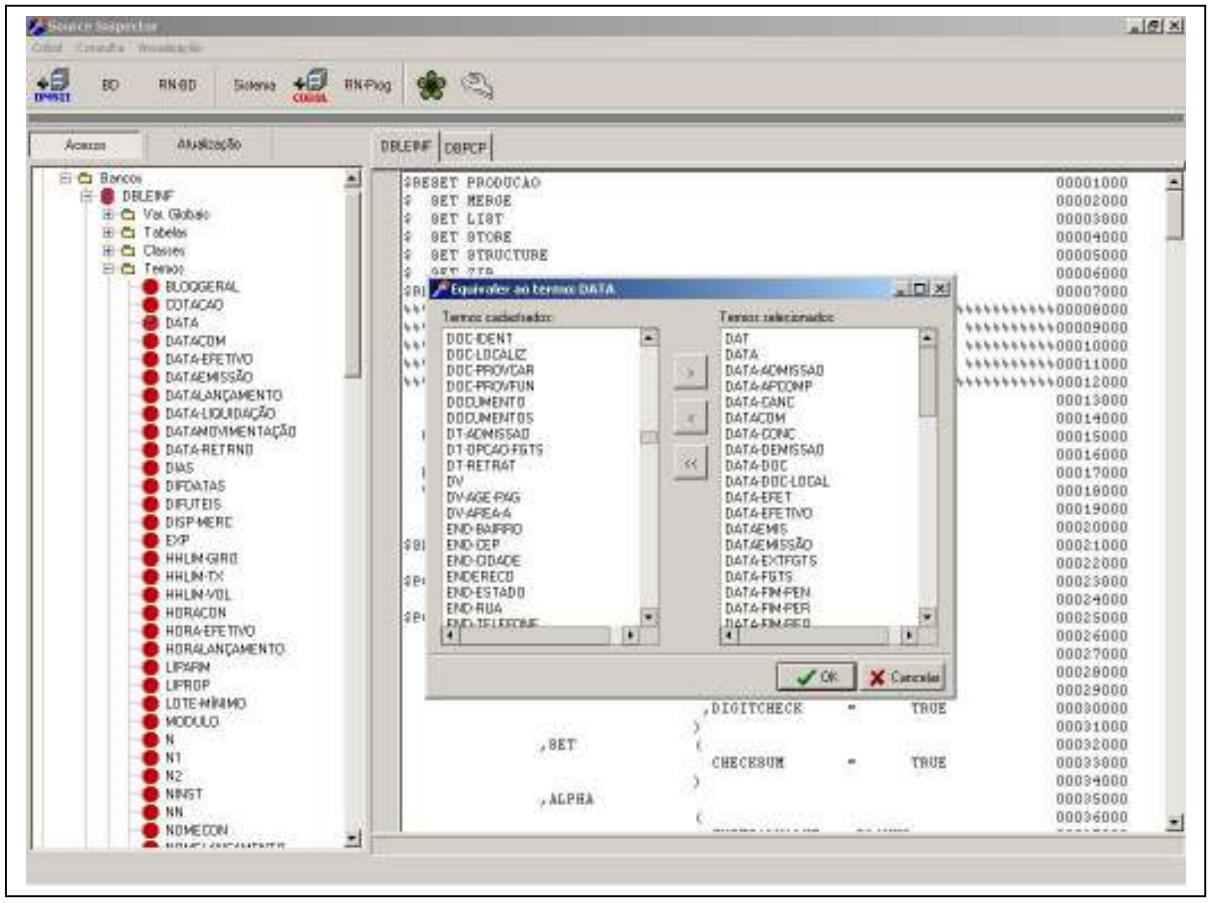

Figura 4.5 – **Interface para a o procedimento de equivalência de termos. Neste momento o especialista pode elencar os termos similares a fim de determinar sua equivalência para o** Source Inspector**. Na simbologia utilizada, o círculo identifica o termo enquanto a cor vermelha classifica o termo como "não acessado" ou "não atualizado", dependendo do folder ativo para a interface (acesso ou atualização).**

4.2 Extração das Regras de Negócio

Os quadros a seguir mostram o resultado da aplicação das heurísticas para cada categoria de regra de negócio no exemplo do sistema Ofpub.

4.2.1 Extraindo Fatos

Heurística	Número de Regras	Comentário
HRNF01	29	Associações do tipo ESTRUT-ITEM. DSOFER (14) e DSPROP (15)
HRNF02	0	Não existem agregações relacionadas.
HRNF03	1	Existe uma Associação ESTRUT-DS, já que DSOFER e DSPROP possuem em comum o termo de negócio PAPEL.
HRNF04	0	Não existem arquivos declarados para o programa em questão
HRNF05	0	Apesar de existir uma associação no programa entre DSOFER e DSPROP, um novo fato não será gerado. Esta informação estará refletida na abrangência do fato, que não será tratada neste nosso exemplo porque estamos tomando uma rotina isolada.
HRNF06	5	Em DSOFER, existem três subsets declarados. Dois deles possuem os mesmos campos na chave. Portanto são gerados apenas dois fatos de ORDENAÇÃO. Em DSPROP, existem quatro subsets declarados. Dois deles possuem os mesmos campos na chave. Desta forma são gerados mais três fatos de ORDENAÇÃO.
HRNF07	0	Não há definição de RECORD TYPE.
HRNF08	0	O termo de negócio PAPEL é definido como identificador em DSOFER (SUBSET com NO DUPLICATES) e depois faz parte de um SUBSET em DSPROP. Entretanto este SUBSET não é do tipo NO DUPLICATES, não sendo portanto identificador.

Quadro 4.11 **– Fatos extraídos pelo** Source Inspector**.**

4.2.2 Extraindo Restrições

Heurística	Número de Regras	Comentário
HRNR01	7	Inicialmente o **Source Inspector** extraiu 7 restrições pois identificou 7 expressões, uma para cada SUBSET declarado. A intervenção do especialista pode reduzir o número de restrições ao avaliar as expressões obtidas. Vale lembrar que o **Source Inspector** não realiza análise de expressões e por isso não pode executar um ajuste fino *per si*.
HRNR02	0	Não foram definidas cláusulas do tipo REQUIRED.
HRNR03	0	Não foram definidas cláusulas do tipo VERIFY.
HRNR04	3	Existem apenas três subsets com cláusula NO DUPLICATES.
HRNR05	0	Não foram identificados parágrafos de leitura.
HRNR06	0	Não foram identificados parágrafos de gravação.
HRNR07	0	Não foram identificados comandos de acesso a banco de dados.
HRNR08	0	Não foram identificados comandos de atualização de banco de dados.

Quadro 4.12 – **Restrições extraídas pelo** Source Inspector**.**

4.2.3 Extraindo Regras das Sentenças

Neste exemplo não temos derivações definidas no banco de dados (cláusula AGGREGATE) nem fatos explicitamente definidos no código. Portanto, para facilitar a visualização, identificaremos as regras ao longo do código.

```
161000*****************************************************************
****
161100 MONTA-DIRETA.
161200*PROCESSA A APURACAO DIRETA
161300*****************************************************************
****
161400
161600           COMPUTE   PRCT   =   (TOTVDA   /   100)   *   PRCDIR-OFER.
Deriv.Cálculo
161700
162000     SET P TO 1.
162100                        PERFORM     RECONSTITUI-TABPRO     THRU
RegHabAção
162200             RECONSTITUI-TABPRO-EXIT UNTIL P > LIM-P.
162300
162500     SET I TO 1.
162600                              PERFORM      LIMPA-TABINS      THRU
RegHabAção
162700             LIMPA-TABINS-EXIT       UNTIL I > LIM-I.
162800
163000     SET  F          TO 1.
163100     SET  P          TO 1.
163200     SET IP          TO P.
163300     MOVE 1          TO LIM-I.
163400     SET  I          TO 1.
163500      MOVE INST-P(P)  TO INST-I(I).                          Deriv.
Simples
163550
163600      MOVE 0           TO TOTACM.                            Deriv.
Inicializ.
163700     MOVE 0          TO TOTFIN.
163800
164000        PERFORM  CORPO-D  THRU  CORPO-D-EXIT  UNTIL  F  >  LIM-F.
RegHabAção
164100
164200 MONTA-DIRETA-EXIT. EXIT.
164300*****************************************************************
****
164400
164500*****************************************************************
****
164600 RECONSTITUI-TABPRO.
164900*****************************************************************
****
165000
```

```
165100      MOVE 0 TO    E-P(P).                                    Deriv.
Inicializ.
165200     MOVE 0 TO QTC-P(P), QTA-P(P).
165300     MOVE 0 TO VFC-P(P), VFA-P(P).
165400     SET P UP BY 1.
165500
165600 RECONSTITUI-TABPRO-EXIT. EXIT.
165700*****************************************************************
****
```

```
165900*****************************************************************
****
166000 CORPO-D.
166200*****************************************************************
****
166300
166500     MOVE 0 TO TOTFXA.
166600
166800     MOVE SALVA TO QPD-F(F).                Deriv. Mud. Domínio
166900
167100                          PERFORM     MONTA-TABELAS-D     THRU
RegHabAção
167200             MONTA-TABELAS-D-EXIT UNTIL P > QP-F(F).
167300
167600           IF  ((TOTACM  +  TOTFXA)  >  TOTVDA)  OR       Prof=2
RegHabAção
167700         ((TOTACM + TOTFXA) = TOTVDA)
167750
167800         COMPUTE RST      =  TOTVDA - TOTACM       Deriv.Cálc.
167900         COMPUTE RT       = (TOTVDA - TOTACM) / TOTFXA
168000         COMPUTE CORTE-PRE ROUNDED = (1 - (RST / TOTFXA)) * 100
168100         COMPUTE CORTE-OFER-WS ROUNDED  =
168200                 (1 - (RST / TOTFXA)) * 10000
168300
168400         PERFORM RATEIA-D THRU RATEIA-D-EXIT UNTIL IP = QP-F(F)
RHA
168450
168500         MOVE RST TO QT-AUX                Deriv. Simples
168550
168600            COMPUTE  VF-AUX     = (QT-AUX  *  PU-P(IP))  /  10000
Deriv.Cálc.
168700         COMPUTE QTD-F(F) = (QTD-F(F) - QTC-P(IP)) + QT-AUX
168800         COMPUTE VFD-F(F) = (VFD-F(F) - VFC-P(IP)) + VF-AUX
168900         SET LIM-D TO F
169000         ADD QTD-F(F) TO TOTACM                Deriv.Cálc.
169050
169100         MOVE TOTACM  TO QTD-F(F)              Deriv.Simples
169150
169200         ADD VFD-F(F) TO TOTFIN                Deriv.Cálc.
169250
169300         MOVE TOTFIN  TO VFD-F(F)              Deriv.Simples
169400         SET F     TO LIM-F
169500
169700     ELSE                                  Prof=2
169800         ADD QTD-F(F) TO TOTACM                Deriv.Cálc
169850
169900         MOVE TOTACM  TO QTD-F(F)              Deriv.Simples
169950
170000         ADD VFD-F(F) TO TOTFIN                Deriv.Cálc
170050
170100         MOVE TOTFIN  TO VFD-F(F).             Deriv.Simples
170200
170300     SET F UP BY 1.                        Prof=1
170400
170500     SET IP TO P.
170600
```

```
170700 CORPO-D-EXIT. EXIT.
170800*****************************************************************
****
```

```
171000*****************************************************************
****
171100 MONTA-TABELAS-D.
171400*****************************************************************
****
171500
171700     SET I TO 1.
171800     MOVE "F" TO ACHOU.
171900              PERFORM   BUSCA-INST-D   THRU   BUSCA-INST-D-EXIT
RegHabAção
172000             UNTIL ((ACHOU = "V") OR (I > LIM-I)).
172100
172300          IF  ACHOU  =  "F"                                    Prof=2
RegHabAção
172400        ADD 1 TO LIM-I
172500        SET I TO LIM-I
172600         MOVE INST-P(P) TO INST-I(I).                         Deriv.
Simples
172700
172900     ADD 1         TO NPP-I(I).                               Deriv.
Cálculo
173000     ADD QTP-P(P) TO QTP-I(I).
173100     ADD VFP-P(P) TO VFP-I(I).
173200
173400          IF  QTA-I(I)  <  PRCT                                Prof=2
RegHabAção
173500
173600         ADD 1 TO QPD-F(F)                                    Deriv.
Cálculo
173700        ADD 1 TO SALVA
173800
173900            IF  (QTA-I(I)  +  QTP-P(P))  NOT  >  PRCT         Prof=3
RegHabAção
173950
174000            MOVE QTP-P(P) TO QTC-P(P)                        Deriv.
Simples
174100           MOVE QTP-P(P) TO QTA-P(P)
174200        ELSE                                          Prof=3
174400             COMPUTE QTC-P(P) = PRCT - QTA-I(I)              Deriv.
Cálculo
174450
174500            MOVE     QTC-P(P) TO QTA-P(P)                    Deriv.
Simples
174600     ELSE                                        Prof=2
174800           MOVE  0  TO  QTC-P(P),  QTA-P(P),  VFC-P(P),  VFA-P(P)
Deriv.Inic
174850
174900        MOVE 2 TO   E-P(P).                                  Deriv. Mud.
Est.
175000
175300       IF  QTC-P(P)  >  0                                    Prof=2
RegHabAção
175350
175400         COMPUTE VFC-P(P) = (QTC-P(P) * PU-P(P)) / 10000   Der.
Cálc
175450
```

```
175500          MOVE VFC-P(P) TO VFA-P(P)                       Deriv.
Simples
175550
175600          ADD  QTC-P(P) TO QTC-I(I)                       Deriv.
Cálculo
175700        ADD  QTA-P(P) TO QTA-I(I), QTD-F(F), TOTFXA
175800        ADD  VFC-P(P) TO VFC-I(I)
175900        ADD  VFA-P(P) TO VFA-I(I), VFD-F(F).
176000
176200      SET II             TO NPP-I(I).        Prof=1      Deriv.
Simples
176300     SET ORDP-I(I,II) TO P.
176400     SET ORDI-P(P)    TO I.
176500     SET ORDII-P(P)   TO II.
176600
176700     SET P UP BY 1.
176800
176900 MONTA-TABELAS-D-EXIT. EXIT.
```

```
177000*================================================================
===*
177100 BUSCA-INST-D.
177400
177500          IF  INST-P(P)  =  INST-I(I)                    Prof=2
RegHabAção
177600        MOVE "V" TO ACHOU
177700     ELSE
177800        SET I UP BY 1.
177900
178000 BUSCA-INST-D-EXIT. EXIT.
178100*****************************************************************
****
178200
178300*****************************************************************
****
178400 RATEIA-D.
178600*****************************************************************
****
178700
178900     COMPUTE QT-AUX = RT * QTA-P(IP).          Deriv. Cálculo
179100       DIVIDE   QT-AUX  BY  WS-MODULO   GIVING  RSLT   REMAINDER
RESTO.
179150
179200      IF RESTO > MDL2 ADD 1 TO RSLT.   Prof=2    RHA / Deriv.
Cálculo
179250
179300      COMPUTE QT-AUX = RSLT * WS-MODULO.      Prof=1     Deriv.
Cálculo
179500     COMPUTE VF-AUX = (QT-AUX * PU-P(IP)) / 10000.
179700     COMPUTE RST = RST - QT-AUX.
180000     COMPUTE QTD-F(F) = (QTD-F(F) - QTC-P(IP)) + QT-AUX.
180100     COMPUTE VFD-F(F) = (VFD-F(F) - VFC-P(IP)) + VF-AUX.
180200
180300     ADD 1 TO IP.
180400
180500 RATEIA-D-EXIT. EXIT.
180600*****************************************************************
****
180700
180800*****************************************************************
****
180900 LIMPA-TABINS.
181100*****************************************************************
****
181200
181300     MOVE 0 TO INST-I(I).                        Deriv Iniciação
181400     MOVE 0 TO  NPP-I(I).
181500     MOVE 0 TO  QTP-I(I).
181600     MOVE 0 TO  VFP-I(I).
181700     MOVE 0 TO  QTC-I(I).
181800     MOVE 0 TO  VFC-I(I).
181900     MOVE 0 TO  QTA-I(I).
182000     MOVE 0 TO  VFA-I(I).
182100     MOVE 0 TO ORDP-I(I,1).
182200     MOVE 0 TO ORDP-I(I,2).
182300     MOVE 0 TO ORDP-I(I,3).
```

```
182400     MOVE 0 TO ORDP-I(I,4).
182500     MOVE 0 TO ORDP-I(I,5).
182600     MOVE 0 TO ORDP-I(I,6).
182700     MOVE 0 TO ORDP-I(I,7).
182800     MOVE 0 TO ORDP-I(I,8).
182900     MOVE 0 TO ORDP-I(I,9).
183000     MOVE 0 TO ORDP-I(I,10).
183100
183200     SET I UP BY 1.
183300
183400 LIMPA-TABINS-EXIT. EXIT.
```

Para entendermos corretamente o que foi assinalado no programa, adotamos os seguintes padrões:

- As linhas sombreadas contêm alguma ou parte de uma regra de negócio;
- Os termos de negócio foram sublinhados;
- A profundidade identificada para o ninho de IFs dentro de cada procedimento foi assinalada (sombreado mais forte – **Prof=x**);

Algumas observações também se fazem necessárias:

- Nem sempre uma regra de negócio está associada a um termo de negócio. No caso das regras habilitadoras de ação e das derivações do tipo cálculo, foram consideradas todas as sentenças. Caberá ao especialista de domínio verificar a relevância desta apuração;

- É importante ressaltar que o ciclo de tempo para se obter resultados a partir do **Source Inspector** depende diretamente do número e do tamanho dos fontes analisados. O tempo de execução tanto no ambiente Unix como no Windows é muito curto e a maior dificuldade reside nas diversas conversões necessárias para migração dos arquivos entre os ambientes (UNISYS para Unix e depois Unix para Windows). Uma outra variante a ser considerada para estimar o tempo é muito aleatória, pois consiste exatamente do tempo gasto pelo especialista para interagir com a ferramenta e refinar as definições para termos e regras de negócio.

- O nível de profundidade assinalado é o que identificamos como profundidade local (no procedimento). Ao observarmos a

estrutura de chamadas, é necessário adicionar o nível de profundidade do procedimento para obtermos a real dimensão sobre o quão "escondida" está determinada regra de negócio ou ação. No exemplo acima, a estrutura de chamada segue o descrito na Figura 4.7, que relaciona a profundidade real do comando no código. Com isso podemos determinar que o nível de profundidade de uma sentença é o resultado da soma do nível de profundidade da chamada com o nível de profundidade do ninho de IF.

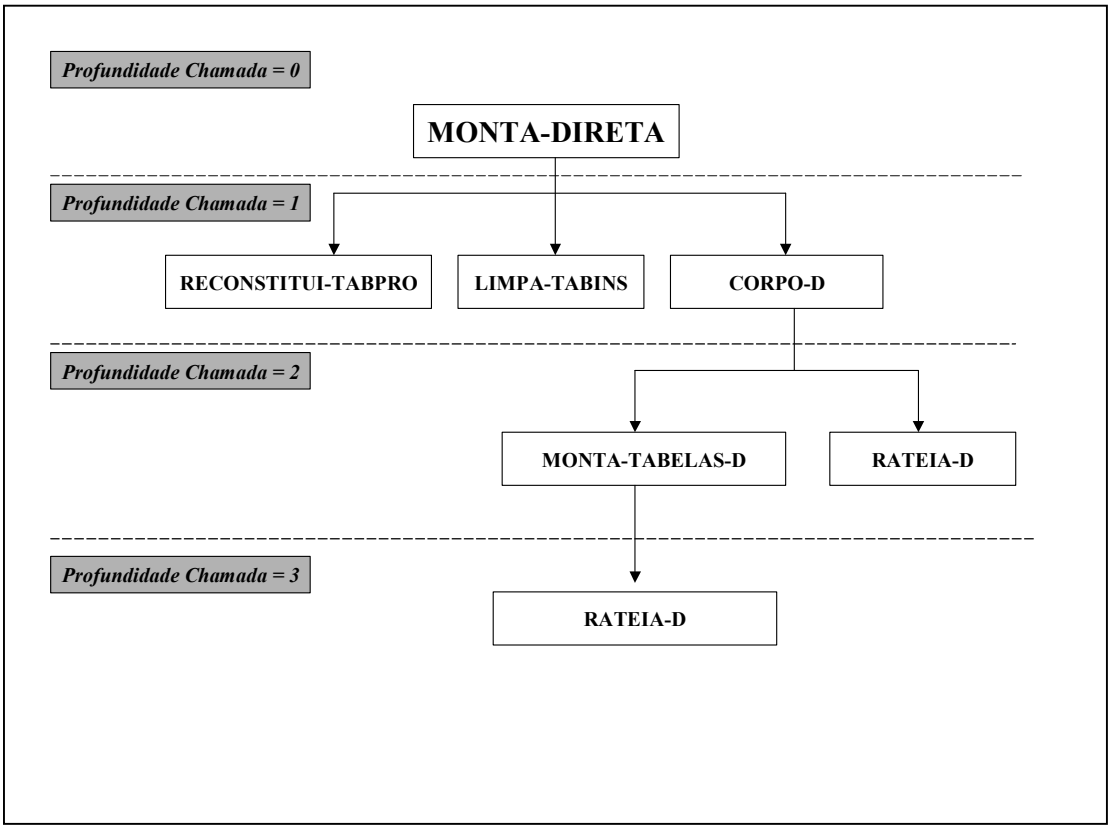

Figura 4.8 **– Profundidade de chamada para os parágrafos.**

4.3 Análise dos Resultados

4.3.1 Regras de negócio extraídas

O processamento do fonte PROCESSA-APURACAO-DIRETA utilizando o banco de dados OFPUB produziu as seguintes regras de negócio:

Regra	**Número**
Termos	60
Fatos	35
Restrições	10
Derivações	32
Regras Habilitadoras de Ação	13
TOTAL	**150**

Quadro 4.13 – **Totalização das regras extraídas.**

A partir deste quadro podemos fazer as seguintes observações:

- O significativo número de termos de negócio representa a influência do modelo estático na determinação das regras. Dos 60 termos, 36 vieram dos datasets, mostrando a importância da dicionarização. Ainda em relação ao modelo estático, é importante observar que boa parte dos fatos gerados representa a relação entre os termos de negócio e sua estrutura (ASSOCIAÇÃO ESTRUT-ITEM);

- Conforme novos programas sejam submetidos ao **Source Inspector** maior a quantidade relativa de regras por sentença, já que as informações sobre o modelo de dados já foram processadas pela ferramenta;

- A geração de uma estrutura de chamadas e da profundidade da sentença, permite um melhor entendimento do código e do seu fluxo de informação. No caso de identificação de ninhos de expressões condicionais, a ajuda do **Source Inspector** torna-se ainda mais decisiva se a codificação não estiver alinhada corretamente conforme as boas práticas de programação;

- As regras habilitadoras de ação podem representar um número elevado para a quantidade reduzida de desvios existente no trecho de código, mas caberá ao especialista de domínio determinar se a regra obtida merece ou não ser considerada como pertencente ao negócio. Os procedimentos RECONSTITUI-TABPRO e LIMPA-TABINS são eminentemente operacionais, já que estabelecem derivações de inicialização para permitir o processamento.

- O especialista de domínio também tem o recurso de reunir sob uma mesma regra um conjunto de regras correlatas. O **Source Inspector** toma a iniciativa ao reunir regras da mesma categoria e do mesmo tipo, mas como podemos observar nos trechos entre as linhas 169000 a 170100, existem oito regras (quatro derivações simples e quatro cálculos) que podem ser sintetizados;

4.3.2 Heurísticas utilizadas

Para identificação das 150 regras foram utilizadas, dentre as 59 heurísticas definidas, as seguintes:

Heurística	Número de Regras
HRNT01	2
HRNT02	36
HRNT03	22
HRNF01	29
HRNF03	1
HRNF06	5
HRNR01	3
HRNR04	7
HRND04	4
HRND06	1
HRND07	14
HRND08	13
HRNH03	8
HRNH04	5

Quadro 4.14 – **Totalização por heurística.**

- Das 59 heurísticas existentes apenas 14 foram efetivamente aplicadas na extração das regras. Devemos levar em conta que para identificação de termos 20 regras são utilizadas para determinar a abrangência (local, programa, sistema, global) e a característica do termo (acesso/atualização). Tal avaliação ainda não foi implementada

na ferramenta, mas em nosso exemplo possui pouca significância dado que estamos trabalhando com uma rotina isolada;

- Sobre as heurísticas relativas à derivação convém considerar que a união de uma seqüência de derivações caracteriza também uma heurística (HRND09). Portanto algumas heurísticas são utilizadas em conjunto para determinar a validade da regra.

O **Source Inspector** possui o grande benefício de propor um modelo de classes inicial que pode servir de base para o projeto de uma nova implementação, independente da plataforma que se irá adotar. Sua grande limitação está na dependência de uma certa estrutura e organização do sistema legado, especialmente na nomenclatura dos campos e itens de dados, fator vital para a identificação de classes e de relacionamentos.

4.3.3 Outros Benefícios do Source Inspector

Um dos pontos interessantes obtidos a partir da análise dos termos é a abrangência no que diz respeito ao acesso e atualização, conforme observamos nas definições das heurísticas (quadro 3.8). Analisando estas informações, podemos ter um interessante resultado que pode servir como insumo para um estudo sobre a matriz CRUD (Create Read Update Delete) dos termos de negócio. Definimos uma simbologia de formas geométricas apropriadas para as regras (termos = círculos, fatos = quadrados, restrições = triângulos, derivações = estrelas, regras habilitadoras de ação = explosão) e uma especificação de cores para cada nível de acesso/atualização (vermelho = não acessado/atualizado, roxo = abrangência local (procedimento); verde = abrangência programa, laranja = abrangência de sistema; azul = abrangência global).

Figura 4.9 – **Simbologia utilizada no** Source Inspector **para caracterizar as regras de negócio na interface.**

A Figura 4.10 abaixo exemplifica uma extração de termos a partir do banco DBLEINF.

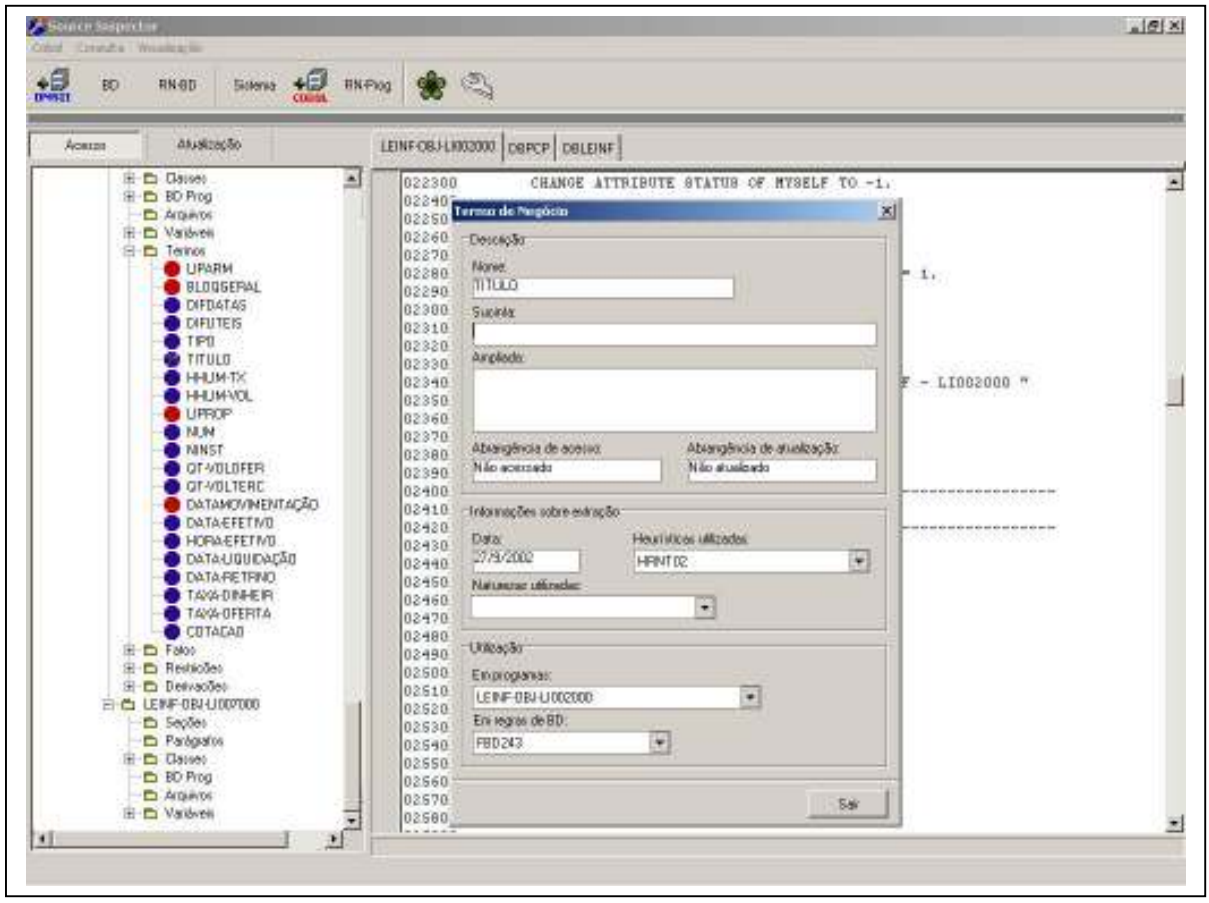

Figura 4.10 – **Termos de negócio extraídos do DBLEINF. Vale observar que há uma interface para edição dos termos por parte do especialista de domínio, de tal forma que a documentação do termo tenha seu nível de abstração elevado.**

Similarmente, os termos utilizados em um determinado programa também são classificados quanto a sua abrangência. Ainda sobre o sistema DBLEINF, segue abaixo o resultado obtido na interface após a identificação dos termos relativos ao programa LEINF-OBJ-LI002000.

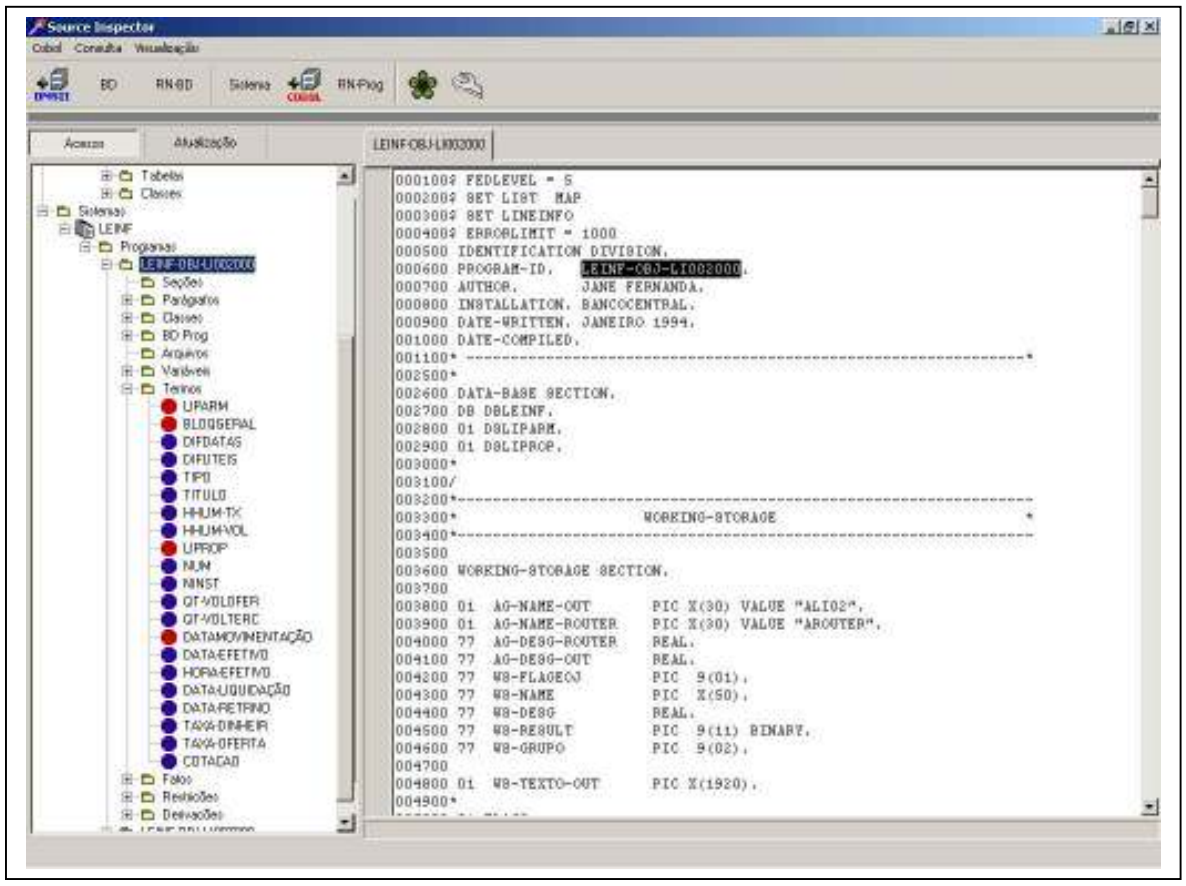

Figura 4.11 – **Termos do programa LEINF-OBJ-002000.**

Em relação à apresentação das regras, o **Source Inspector** pode apresentar as regras habilitadoras de ação de uma forma hierárquica. Ou seja, as regras extraídas são seqüências do tipo *condição-ação* [SULA99], onde cada ação pode desdobrar-se em um outro conjunto de regras, ações e condições até que seja atingida a atomicidade da regra. Esta abordagem permite que o **Source Inspector** possa ser utilizado para diversos fins. Desde uma ferramenta para aqueles que desejam apenas um conhecimento superficial do aplicativo (ou seja, sua seqüência de parágrafos e rotinas) até como auxílio para os programadores que precisam de informações mais detalhadas sobre o código existente.

Esta analogia existe também para o nível de representação das regras de negócio. Executivos esperam um nível maior de abrangência e representação, os

usuários de um sistema esperam outro. Os mantenedores do software por sua vez esperam uma outra representação, com mais detalhes e mais próxima da codificação utilizada a fim de que a tarefa de manutenção possa ser realizada mais facilmente. [HUAN98]

Nossa proposta não chega a abranger representações diferentes, mas apenas níveis de detalhe diferentes tendo como base o fonte COBOL. O recurso existente no **Source Inspector** para promover tal compreensão do legado é a profundidade. Existem dois tipos de profundidade:

- Profundidade de aninhamento – São os ninhos de IFs que os programas legados costumam apresentar.

- Profundidade de PERFORM - Este tipo de profundidade determina as chamadas que cada seção/parágrafo faz, dando um mapeamento da estrutura modular do código analisado. Cada chamada corresponde a um outro grupo de comandos (ações e regras).

A proposta do **Source Inspector** é permitir a visualização e classificação das regras verificando a quantidade de aninhamento de condições e de detalhamento de ações até que se chegue a alguma regra indivisível.

Além de facilitar a compreensão do código observamos uma oportunidade para validar sua adequação a padrões parametrizados durante o processo de identificação de termos. Obviamente, quanto mais organizado, estruturado e dentro dos padrões cadastrados estiver o programa, mais fácil será sua compreensão.

Um dos resultados adicionais do **Source Inspector**, além da extração de regras de negócio, é a indicação sobre a qualidade do código legado de acordo com alguns parâmetros de qualidade informados pelo especialista, tais como:

- Profundidade máxima dos ninhos de IFs.
    - em um módulo (Recomendado = 3)
    - no programa todo
- Número máximo de comandos GO-TO
    - GO TO DEPENDING;
    - GO TO que implementem iterações;
- Loops iterativos que são controlados externamente (PERFORM VARYING);
- Número máximo de linhas aceitável para um parágrafo;
- Organização dos parágrafos
    - Existe numeração e ela é respeitada;
    - Parágrafos chamados estão sempre abaixo dos chamadores;
- Tamanho efetivo do programa
    - Número de módulos (parágrafos)
    - Seqüência de chamadas (profundidade dos módulos) recorrentes (acumulando)
    - Número de chamadas (máximo possível)
    - Número real de chamadas / número de módulos.
- *Dead code* (rotinas que não são executadas);
- Itens de dados não utilizados pelo sistema;
- Número de linhas por divisão do COBOL.

5 CONCLUSÃO

"The reward for a well done job is
the chance to do more well done job."
Jonas Edward Salk – microbiólogo americano

"Instead of seeing the butterfly emerge in one day
from the cocoon, you plan its metamofphosis and
track its evolution iteration by iteration"
Joe Marasco, da Rational, sobre migração de sistemas legados

5.1 Contribuições

1. A grande contribuição deste trabalho está na definição das heurísticas para identificação e classificação de regras de negócio. O registro destas heurísticas explicita um conhecimento sobre a interpretação de fontes COBOL que pode ser constantemente refinado conforme novos especialistas transmitam suas contribuições. Na implementação percebemos que a força das heurísticas de regras de negócio está diretamente relacionada ao grau de eficiência na parametrização informada pelo especialista de domínio e ao grau de organização do código legado. Neste ponto, o **Source Inspector** premia aqueles que, mesmo em ambientes áridos para o desenvolvimento de sistemas e para a engenharia de software lograram algum esforço qualitativo. As heurísticas adotam também um princípio de conservadorismo nos trechos de código mais duvidosos em relação a uma interpretação automático. Desta forma, evitam-se possíveis intervenções incorretas no código sem a participação do especialista e com resultados questionáveis;

2. Conforme foi possível verificar existe um processo claramente interativo para extração de regras de negócio. Este processo desenvolvido numa abordagem centrada nos dados é implementado pelo **Source Inspector**;

3. Os resultados fornecidos pelas análises realizadas permitem que tenhamos informações importantes para a compreensão do código legado e uma possível documentação dos programas. Além disso, estas análises podem ser utilizadas como instrumento para tomada de decisão sobre onde e como alterar o fonte, determinando que partes do programa podem ser melhoradas;

4. A iniciativa do **Source Inspector** representa uma alternativa simples para classificação de termos e regras de negócio que pode ser desenvolvida para taxonomias mais complexas ou mesmo customizada se for conveniente para a organização;

5. A abordagem do **Source Inspector** é bastante particular, ao contrário de outras linhas que propõem ferramentas de engenharia reversa genéricas. [VERH00, TONE00] Neste caso estamos atacando necessidades específicas dos clientes de ambiente UNISYS / COBOL /DMSII e ainda permitindo a parametrização de uma série de itens de qualidade determinados pelos próprios especialistas de domínio, incluindo padrões de busca e identificação das regras de negócio;

6. Para a gestão de conhecimento o **Source Inspector** também apresenta sua contribuição na medida em que melhora sobremaneira o entendimento de um sistema de negócio que se apresentava totalmente codificado e que a partir da aplicação de algumas heurísticas pode ser mais facilmente compreendido. Esta compreensão pode ser explícita na catalogação dos termos de negócio

obtidos (em última análise uma dicionarização) e na classificação das RN.

7. Outro aspecto importante do trabalho foi fazer uma proposta de extração de regras em ambientes legados que pode integrar-se a outras propostas de representação de regras em um nível mais elevado. Desta forma os metamodelos de regras desenvolvidos a partir da especificação de mais alto nível podem ser refinados pelas regras extraídas;

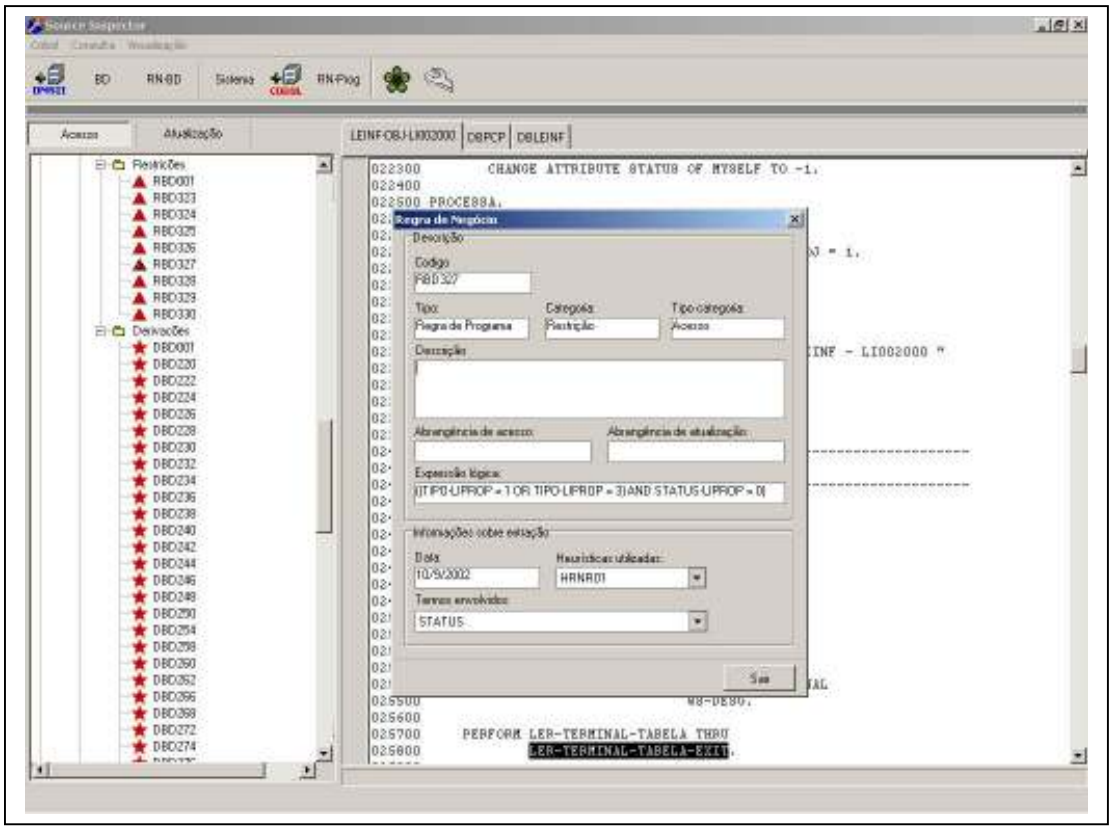

Figura 5.1 – **Interface do** Source Inspector **com a tela para edição de documentação das regras. Este cadastro pode ser o ponto de partida para integração com outras ferramentas de representação em mais alto nível.**

8. O trabalho de implementação também contribuiu para que o **Source Inspector** fosse transformado efetivamente em um produto. Além da implementação das heurísticas de regras de negócio, houve um grande esforço para tornar o **Source Inspector** uma ferramenta robusta o suficiente para que não fosse necessário alterar uma única linha de código dos fontes legados para evitar algum erro de interpretação por parte dos analisadores. Este esforço permitiu que códigos mais complexos e efetivamente de sistemas em produção, como os do Banco Central e os da Dataprev pudessem ser tratados sem maiores problemas;

9. Especialmente no Banco Central, patrocinador deste trabalho de dissertação, o **Source Inspector** pode perfeitamente ser incorporado à nova metodologia de desenvolvimento de sistemas adotada pelo Departamento de Informática (Deinf). Esta metodologia utiliza o RUP (Rational Unified Process) e, conforme descrito em [KRUC01] os princípios do RUP (desenvolvimento integrado, acompanhamento baseado em medições, verificação contínua de qualidade, gerenciamento de escopo, etc.) podem ser aplicados ao desenvolvimento de qualquer tipo de software. Seu *template* de ciclo de vida, com as fases de Concepção (Iniciação), Elaboração, Construção e Transição, é plenamente aplicável a projeto que trate da evolução de um sistema legado. [TOUR02]

5.2 Trabalhos Futuros

- Não encontramos na literatura propostas muito claras para a medição das regras de negócio, abrangendo aspectos como freqüência de utilização,

importância da regra no contexto do negócio, abrangência, etc. Ao propor uma taxonomia simplificada, o **Source Inspector** pode contribuir como ponto de partida para uma abordagem deste tipo;

- Em termos de implementação, podemos relacionar uma série de melhorias que seriam interessantes, dentre as quais destacamos:
    1. Geração dos analisadores léxico e sintático no ambiente NT ou no Linux, de tal forma que a interface com estes analisadores seja melhorada. Atualmente existem versões interessantes do Lex e do Yacc em outros ambientes. Evitamos utilizá-las porque nosso foco não era especificamente verificar as diferenças entre ambientes, mas sim viabilizar a implementação das heurísticas. Este estudo poderia melhorar a interação com o produto na medida em que não seria mais necessário dispor do ambiente Unix/Solaris. Todo o aplicativo poderia residir sob a mesma plataforma (Windows) facilitando sua manutenção;
    2. O Módulo Extrator (ModEx) desenvolvido em Delphi poderia ter sua interface remodelada para o padrão Web, tornando o produto ainda mais amigável e facilmente implementável em organizações mais complexas e dispersas;
    3. O analisador COBOL por nós desenvolvido poderia ser preparado para reconhecer as informações produzidas pelo próprio compilador COBOL. Atualmente trabalhamos com os fontes puros, mas o resultado da compilação de fontes no ambiente UNISYS traz muitas informações interessantes, como referência cruzada de variáveis, listagem de procedimentos, etc;
    4. Outro recurso interessante para o analisador COBOL que não foi implementado nesta versão é a interpretação de bibliotecas que sejam invocadas ao longo do código (comando COPY). Esta implementação agregaria muito valor por permitir que a

análise de um único fonte seja capaz de identificar o conteúdo das bibliotecas por ele utilizadas.

5. Para implementação em ambientes organizacionais seria recomendável a conversão das bases de dados do ambiente local (Paradox) para um ambiente mais corporativo com SGBD relacional de maior porte, como Oracle, Sybase, etc.
6. O tratamento que o **Source Inspector** faz para classes e atributos poderia também gerar esquemas automáticos para ambientes relacionais. Estes esquemas podem representar uma primeira abordagem para um processo de migração de plataformas, quando fosse necessário;
7. Especificamente ainda em relação ao ambiente UNISYS, convém citar que os fluxos de processamento no ambiente de grande porte são definidos por *Jobs*. Estes utilizam o WFL (Work Flow Language), uma linguagem que define os fluxos de execução dos programas. O desenvolvimento de um analisador específico para compreensão dos fontes WFL agregará importantes informações sobre os sistemas legados que podem ser integrados ao **Source Inspector**;
8. Apesar de fortemente voltado para o ambiente UNISYS, nada impede que os analisadores possam, com pequenas alterações, interpretar corretamente o código COBOL de outros ambientes (IBM, Fujitsu, ABC BULL). Além disso, muitas das heurísticas para obtenção de termos de negócio e algumas para obtenção de regras podem ser adaptadas para o modelo relacional;
9. Versões mais recentes do COBOL trazem recursos mais poderosos para utilização de funções pré-definidas. Tais recursos exigirão do analisador maior poder para interpretar funções e parâmetros, o que poderia auxiliar ainda mais na análise do fonte e na extração de regras. [SCHM88]

- O **Source Inspector** pode integrar-se a um ambiente colaborativo de conhecimento. Ele permite a interação entre o especialista de domínio, o analista responsável pelo sistema e o analista de negócios. Essas interações poderiam ser registradas em um software específico para este fim, envolvendo questões como segurança, certificação e identificação dos usuários (no caso especialistas), contribuindo assim consideravelmente para uma melhor compreensão do legado;

- Ao fazer uma abordagem partindo da linguagem procedural, o **Source Inspector** pode interagir, através de seu metamodelo, com ferramentas que trabalhem com o armazenamento de regras declarativas, permitindo assim outras formas de representação que não fossem simplesmente as linhas de código. O grande benefício desta sinergia seria a obtenção de uma representação de mais alto nível das regras, como por exemplo lógica de primeira ordem. [CORR02, GHEZ91] O grupo de pesquisa do professor Eber Schmitz está trabalhando no desenvolvimento de um ambiente denominado *Modelador de Empresas*. Baseado nos conceitos descritos em [ERIK00], podemos inferir que o processo de desenvolvimento de sistemas está baseado no tripé *objetivos, processos* e *regras*. O **Source Inspector** pode juntamente com o *RuleCheck* [CORR02], atuar tanto na parte de regras como de processos, fornecendo insumos para o *Modelador de Sistemas de Informação*, conforme apresentado na figura abaixo:

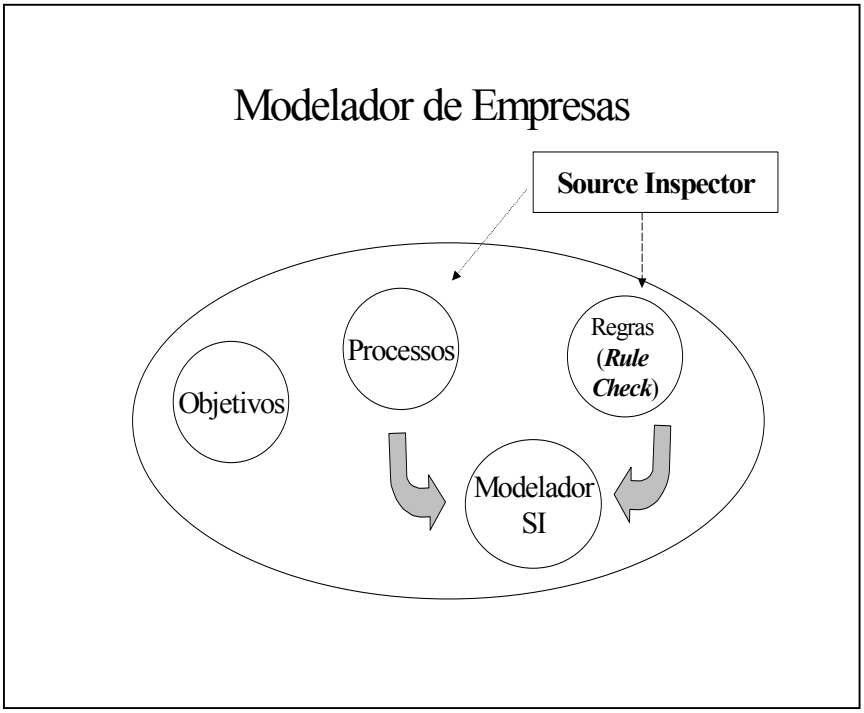

Figura 5.2 – **Modelador de Empresas.com participação do** Source Inspector

- Não fizemos um aprofundamento para integração de métodos e regras, O desenvolvimento de uma solução, utilizando o diagrama de atividades da UML, permitiria a integração do **Source Inspector** à UML e a outras ferramentas que auxiliassem no processo de desenvolvimento de software, como o Rational Rose, por exemplo [RATI98]. Uma das alternativas neste sentido é considerar cada programa como uma classe de controle. Cada parágrafo por sua vez, passa a ser um método desta classe. A partir daí um mapeamento pode ser construído com as classes propostas (baseada no modelo de dados e na estrutura de arquivos dos programas) para estabelecermos as associações entre as classes de atributos utilizadas e a classe de controle em questão;

- Nas funções para identificação de termos de negócio e sua relação com variáveis de programa e outros itens de dado, pode ser realizado um

trabalho mais profundo utilizando-se técnicas mais elaboradas de *slicing*.[HUAN98, ELSE98] Neste trabalho não adotamos esta abordagem por considerarmos que este recurso está muito mais voltado para um ambiente de manutenção de aplicativos do que propriamente para extração de regras;

- Na questão de extração de regras sobre ambiente legado, existem outras abordagens complementares ao **Source Inspector** que podem ser utilizadas para melhoria do conhecimento deste patrimônio de software. Uma das abordagens foi descrita em [SULA00] e aplicada em ambiente UNISYS para identificação de seqüências de regras endógenas nas transações com bancos de dados a partir dos logs gerados pelo DMSII [SULA02];

- Outro aspecto importante no estudo de regras de negócio consiste da avaliação de sua consistência. Isso significa responder perguntas do tipo: "As regras obtidas em dois programas diferentes são conflitantes?", "Regras semelhantes em programas diferentes podem ser consideradas como uma só?". Estas questões requerem um estudo particular mas cujo passo inicial foi dado pelo **Source Inspector**, mesmo este apresentando a limitação de não interpretar as expressões, mas apenas considerá-las no contexto das regras.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[AHO86] AHO, Alfred V., SETHI, Ravi and ULLMAN, Jeffrey D. *Compilers – Principles, Techniques and Tools*. Addison-Wesley – 1986

[ALEN02] ALENQUER, Pablo L. *Regras de negócio para análise em ambientes OLAP*. Dissertação de Mestrado – DCC/IMM-NCE/UFRJ – Julho de 2002.

[BCB95] BANCO CENTRAL DO BRASIL, *Legislação Básica do* **SELIC**, *MNI 6.3*, 1995

[BCB96] BANCO CENTRAL DO BRASIL, *Economia para não-economistas*, Vol. 3, 1996.

[BCB97] BANCO CENTRAL DO BRASIL, *Programa de capacitação para analista e procurador do Banco Central*, Modulo IV, 1997.

[BCB98] BANCO CENTRAL DO BRASIL, *Manual do Usuário do* **SELIC**, 1998.

[BENN96] BENNETT, Keith. *Software evolution: past, present and future.* Information and Software Technology, vol. 38, no. 11, p.673-680, Elsevier, NL, 1996.

[BRAN96] BRAND, Mark Van der, KLINT, P. and VERHOEF, Chris. *Reverse engineering and system renovation – an annotated bibliography.* Technical Report, Programming Research Group, University of Amsterdam, Jul/1996, disponível na web em http://adam.wins.uva.nl/~x/reeng/REanno.html em 29/09/2002.

[BRAN97] BRAND, Mark Van der; SELLINK, Alex and VERHOEF, Chris. *Obtaining a COBOL grammar from legacy code for reengineering purposes*. Technical Report, Programming Research Group, University of Amsterdam, Setembro de 1997, disponível na web em http://adam.wins.uva.nl/~x/Coboldef/Coboldef.html em 29/09/2002.

[BROD95] BRODLE, Michael and STONEBRAKER, Michael. *Migrating legacy systems*. San Francisco: Morgan Kaufmann Publishing, 1995.

[BUSI99] BUSINESS Rules Group, *What is a business rule?*, 1999. Disponível na web em http://www.businessrulesgroup.org/brgdefn.htm em 29/09/2002.

[CANT98] CANTÙ, Marco. *Dominando o Delphi 4 – A Bíblia*. Pág. 190-198. Ed. Makron Books, São Paulo, 1998.

[CHAP02] CHAPIN, Donald. *Rule: is it a primitve or a composite?*. Business Rules Journal, Vol. 3, No. 2, Fev/2002, disponível na web em http://www.brcommunity.com/a2002/b099.html em 29/09/2002.

[CHAP96] CHAPIN, Ned and LAU, Tony S. *Effective size: an example of use from legacy systems*. Software Maintenance: Research and Practice, vol. 8, p. 101-116, John Wiley & Sons, Ltd., 1996.

[CORB89] CORB, T. A. *Program understanding: challenge for the 1990's*. IBM System Journal 28(2), p.294-306, 1989.

[CORR02] CORRÊA, Sergio; SIQUEIRA, José R. B., SCHMITZ, Eber. *Representação de regras de negócio em lógica de primeira ordem*. XXXIV Simpósio Brasileiro de Pesquisa Operacional, 2002.

[CURT89] CURTIS, B. *et al. Experimental evaluation of documentation formats*. Journal of System and Software, 9(2), 167-207.

[DATE00a] DATE, C. J. *What not how*: *the business rules approach to application development*, 1ª ed, Addison-Wesley, Massachussets, Apr/2000.

[DATE00b] DATE, C.J. *An introduction to database systems*. 7ª ed, Addison-Wesley, Massachussets, 2000.

[DATE00c] DATE, C.J. *Introdução a sistemas de bancos de dados*. 7ª ed, Ed. Campus, p.154, 2000.

[DEUR99] DEURSEN, Arie van, Paul Klint and Chris Verhoef. *Research issue in the renovation of legacy systems*, University of Amsterdam, Programming Research Group, 1999, disponível na Web em http://adam.wins.uva.nl/~x/etaps/etaps99.html em 29/09/2002.

[ELMA94] ELMASRI, R. and NAVATHE, S. *Fundamentals of Database Systems*, 3rded., p.484, Addison-Wesley, Reading Massachussets, 1994.

[ELSE98] ELSEVIER. *Special Issue – program slicing*. Information and Software Technology, vol. 40, números 11-12, pp.575-730, 1998.

[ERIK00] ERIKSSON, Hans-Erik; PENKER, Magnus. *Business modelling with UML – business patterns at work*. OMG Press, John Wiley & Sons, Inc. 2000.

[FERR86] FERREIRA, Aurelio Buarque de Holanda *et al.* *Novo Dicionário da Língua Portuguesa*, p. 1475, Ed. Nova Fronteira, 1986.

[FREI97] FREITAS, Felipe Gouveia e LEITE, Julio Cesar Sampaio do Prado. *Aplicando reuso de software na construção de ferramentas de engenharia reversa*. Anais do XI Simpósio Brasileiro de Engenharia de Software, Sociedade Brasileira de Computação, Marcos Borges (editor), pp. 265-280. Out/1997.

[GASK00] GASKIN, James E. *COBOL experts: live after Y2K*, 20/01/2000, disponível na web em http://www.mvshelp.com/COBOL_story01.htm em 29/09/2002.

[GAVI00] GAVINHO, Ana de Fátima Paulain. *Extração de um modelo de classes de um sistema COBOL e banco de dados DMS II.* Dissertação de Mestrado – NCE/UFRJ – Março de 2000.

[GHEZ91] GHEZZI, Carlo; JAZAYERI, Mehdi and MANDRIOLI, Dino. *Fundamentals of software engineering*, Prentice-Hall International Editions, New Jersey, p.199-226, 1991

[GOTT00] GOTTESDIENER, Ellen. *Business rules : show power, promise*, Application Development Trends, vol. 4, no. 3, Mar/1997.

[GUID97] GUIDE Business Rules Project: *Final Report*, revisão1.2, Out./1997.

[HALL01a] HALLE, Barbara von, Building rules systems, *DM Review*, Jan-Apr. 2001.

[HALL01b] HALLE, Barbara von, *Business rules applied: building better systems using the business rule approach*, John Wiley & Sons, Inc., 1st ed, 15/09/2001.

[HAMM01] HAMMER, Michael and CHAMPY, James. *Reengineering the corporation: a manifesto for business revolution*. Harper Business, 05/06/2001.

[HAMM02] HAMMER, Michael. *Reengineering work: don´t automate, obliterate*. Harvard Business Review, Haravard Business School Press, 14/01/2002.

[HAY01] HAY, David C. *Concepts, logic, and the Zachman framework*. Curmudgeon's Corner, Feb./2001. Disponível na Web em

http://www.brcommunity.com/cgi-bin/x.pl/features/b052.html em 29/09/2002.

[HAY99] HAY, David C. *Managing business by the rules*. Essential Strategies, Inc, 1999, disponível da Web em http://essentialstrategies.com/publications/businessrules/bruleseco.htm em 29/09/2002.

[HEIM00] HEIMAN, Richard V. *Transforming legacy software assets for e-business: the SEEC approach*, White Paper, International Data Corporation, 2000.

[HSMA00] STORCK, John and Patricia A. Hill. *Knowledge diffusion through strategic communities*, Sloan Management Review, jan.-feb./ 2000 *in* HSM Management, nº 22, sep.-oct. 2000, pág 59-83.

[HUAN98] HUANG, Hai, *et al*. *Business rule extraction techniques for COBOL programs*. Journal of Software Maintenance: Research and Practice, vol. 10, 04/12/1998, pp. 3-35.

[INTE00] INTERCOMP, *MINEIT:Automated extraction of business rules from legacy COBOL applications to Java objects and Enterprise Java Beans*, White Paper, Sep./2000,

[KERN86] KERNIGHAN, Brian W. and RITCHIE, M. *C – A linguagem de programação.* Ed. Campus, Rio de Janeiro, 1986.

[KIM02] KIM, Young S. *What's COBOL?*. Aboutlegacycoding publication, Vol 4, Issue 3, Jun/2002.

[KRUC01] KRUCHTEN, Philippe. *Using the RUP to evolve a legacy system*. The Rational Edge, Mai/2001, disponível na web em http://www.therationaledge.com/content/may_01/t_legacy_pk.html em 29/09/2002.

[LAM00] LAM, Gladys S. W. *et al*. *Business Rules Forum 2000 Practitioner's panel – the dos and don'ts of business rules*. Oct./2000, Atlanta, Georgia, USA, disponível na Web em BRForum-2000 Practitioner's Panel, em http://www.businessrulesforum.com/BRF2000_panel.htm em 29/09/2002.

[LAM02] LAM, Gladys S. W. *et al*. *Business Rules Forum 2001 Practitioner's panel – the dos and don'ts of business rules*. Business Rules Journal, Vol. 3, No. 2, Fev/2002, disponível na Web em http://www.brcommunity.com/a2002/b092.html em 29/09/2002.

[LEIT95] LEITE, Julio Cesar Sampaio do Prado, SANT´ANNA, Marcelo PRADO, Antonio Francisco do. *Porting COBOL programs using a transformational approach*. Anais do IX Simpósio Brasileiro de Engenharia de Software, SBC, pp. 397-413, Out/1995.

[LEVI95] LEVINE, John R., MASON, Tony and BROWN, Doug. *Lex & yacc*. 2nd. Ed, O'Reilly, USA, Feb./1995.

[NETR00] NETRON Corporation. *Assetminer extends legacy systems*. Software Development Times, 01/08/2000, disponível na web em http://www.netron.com em 13/02/2001.

[NING94] NING, Jim Q., ENGBERTS, Andre e KOZACZYNSKI W. *Automated support for legacy code understanding*. Communications of the ACM, Vol. 37, no. 5, pp.50-57, May-1994.

[ÖZSU99] ÖZSU, M. Tamer and VALDURIEZ, Patrick. *Principles of distributed database systems,* 2nd ed, p. 283, 1999.

[PAUL99] PAUL, Lauren Gibbons. *Forever COBOL*, Datamation from earthweb, Junho/1999, disponível na Web em http://itmanagement.earthweb.com/entdev/article/0,,11979_611261,00.html em 29/09/2002.

[PEAR85] PEARL, Judea. *Heuristics: intelligent search strategies for computer problem solving.* Addison-Wesley, Los Angeles-California-USA, October/1985.

[PERK02] PERKINS, Alan, *Business rules are metadata*, Business Rules Journal, Vol. 3, No. 1, Jan/2002, disponível na web em http://www.brcommunity.com/a2002/b097.html em 29/09/2002.

[PRES97] PRESSMAN, Roger S. *Software engineering – a practitioner's approach*, 4ª Ed, McGraw Hill, 1997.

[PROG99] PROGENI, Corporation. *1Spec and the IBM COBOL environment*, Out-Dez/1999, disponível na Web em http://www.progeni.com em 29/09/2002.

[RATI98] RATIONAL Unified Process. *Guidelines: Reverse-engineering Relational Databeses,* Rational Software Corporation – 1998

[ROSS97] ROSS. Ronald G. *The business rules book: classifying, defining, and modeling rules*, 2ª ed., Business Rules Solution, Inc. Houston, Texas : Addison-Wesley, Feb/1997.

[ROSS98] ROSS, Ronald G. *Business rule concepts*, *the new mechanics of business information systems*, Business Rules Solution, Inc., DataToKnowledge Newsletter, Houston Texas, 1998.

[ROSS00] ROSS, Ronald G. *What is a 'business rule'?* Mar/2000, disponível na Web em http://www.brcommunity.com/cgi-bin/x.pl/commentary/b005.html em 29/09/2002.

[ROSS02] ROSS, Ronald G. *What does it mean to be business-driven? (Part 2)*. Business Rules Journal, Vol. 3, No. 7, Jul/2002, disponível na web http://www.brcommunity.com/a2002/b103b.html em 29/09/2002.

[SELL97] SELLINK, Alex and VERHOEF, Chris. *Reflections on the evolution of COBOL*, University of Amsterdam, Programming Research Group , 27/11/1997, disponível na Web em http://adam.wins.uva.nl/~x/lib/lib.html#126 em 29/09/2002.

[SAYL96] SAYLES, Jonathan. *COBOL and the business program paradigm*, MicroFocus, 1996, obtido da Web em http://www.microfocus.com em 30/04/2000.

[SCHI96] SCHILDT, Herbert. *C completo e total*, 3ª ed., Makron Books, São Paulo, 1996.

[SCHM88] SCHMITZ, Eber; TELES, Antônio Aníbal de Souza. *Pascal e técnicas de programação*. 3ª Ed, LTC, 1988.

[SEEC00] SEEC, Inc. *Shaping e-legacies into e-business*, 2000, White Paper, obtido da Web em 30/04/2000 em http://www.seec.com

[SELL97] SELLINK, Alex and Chris Verhoef. *Reflections on the evolution of COBOL*, University of Amsterdam, Programming Research Group, 27/11/1997, disponível na Web em http://adam.wins.uva.nl/~x/lib/lib.html em 29/09/2002.

[SIQU00] SIQUEIRA, José Roberto Brasileiro de; SCHMITZ, Eber. *Heurísticas para identificação e classificação de regras de negócio em programas* COBOL. XXXII Simpósio Brasileiro de Pesquisa Operacional, Universidade Federal de Viçosa, Viçosa-MG. 20/10/2001

[SIQU01] SIQUEIRA, José Roberto Brasileiro de; SCHMITZ, Eber. *Extração de regras de negócio em sistemas legados : uma revisão*. IV Simpósio de Pesquisa Operacional e Logística da Marinha, Escola de Guerra Naval, Rio de Janeiro-RJ, 19/12/2001

[SIQU02] SIQUEIRA, José Roberto Brasileiro de; NATIVIDADE, Luis Campos da; SCHMITZ, Eber. *Source Inspector: uma ferramenta para extração de regras de negócio em sistemas legados*. International Symposium on Knowledge Management/Document Management, PUC/PR, Curitiba, Ago/2002

[STRO99] STROULIA E., *et al.* *Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach*. About Legacy Coding, Article Archives, Vol. 2, Issue 2, Abril/2000, Computing Science Department, University of Alberta – Canada, disponível na Web em http://www.aboutlegacycoding.com/Articles/V20207.asp em 29/09/2002.

[SULA00] SULAIMAN, Alberto *et al, Tecnologia da Informação para a gestão do conhecimento*: regras de negócio, International Symposium on Knowledge Management/Document Management, PUC/PR, Curitiba, Nov./2000.

[SULA02] SULAIMAN, Alberto. *Mineração de dados endógenos*. Tese de Doutorado – COPPE/UFRJ – Rio de Janeiro-RJ, Maio/2002.

[SULA99] SULAIMAN, Alberto *et al. Gerenciando Regras de Negócio: Estado Atual de um Projeto*. Developers Magazine – Novembro/1999;

[TEIX00] TEIXEIRA, Steve; PACHECO, Xavier. *Delphi 5 – Guia do Desenvolvedor*. Ed. Campus, Rio de Janeiro, 2000.

[TONE00] TONELLA, P., *et al. Reverse engineering 4.7 million lines of code*. Software Practice and Experience, 2000; v. 30, pp 129-150, John Wiley & Sons, Ltd.

[TOUR02] TOURNIER, Christophe. *Modelling guidelines for legacy applications*. The Rational Edge, SET/2002, disponível na web em http://www.therationaledge.com/content/sep_02/t_modellingGuidelines_ct.jsp em 29/09/2002.

[UNIS96a] UNISYS Corporation, *ClearPath HMP NX and A Series COBOL ANSI-74 Programming Reference Manual : Basic Implementation (8600-0296-201)*, vol. 1, Junho/96.

[UNIS96b] UNISYS Corporation, *ClearPath HMP NX and A Series COBOL ANSI-74 Programming Reference Manual : Product Interfaces (8600-0130-301)*, vol. 2, Junho/96.

[UNIS96c] UNISYS Corporation, *ClearPath HMP NX and A Series COBOL ANSI-85 Programming Reference Manual : Product Interfaces (8600-1526-201)*, vol. 2, Junho/96.

[UNIS97a] UNISYS Corporation, *ClearPath HMP NX and A Series COBOL ANSI-85 Programming Reference Manual : Basic Implementation (8600-1518-302)*, vol.1, Fevereiro/97.

[UNIS97b] UNISYS Corporation, *ClearPath HMP NX and A Series DMSII Data and Structure Definition Language (DASDL) Programming Reference Manual : Product Interfaces (8600-0213-403)*, vol.2, Setembro/97.

[VERH00] VERHOEF, Chris. *The realities of large software portfolios*, University of Amsterdam, Programming Research Group, 24/02/2000, disponível na Web em http://adam.wins.uva.nl/~x/lsp/lsp.html em 29/09/2002.

[VILA97] VILANOVA, Luciana de Oliveira. *Reengenharia para Reutilização de Software: Uma abordagem para Recuperação de Projetos Orientados a Objeto*. Dissertação de Mestrado, COPPE/UFRJ, Março de 1997

[ZACH87] ZACHMAN, John A. A framework for information systems architecture, *IBM Systems Journal*, vol. 26, n. 3, 1987, pp. 276-292.

[ZACH00] ZACHMAN, John. *Business rules and the enterprise architecture*. Discussão extraída do evento Business Rules Forum, realizado em 27/10/2000, Atlanta, Gorgia, USA, sessão Birds of a feather session, transcrita por Cindy Scullion, disponível na Web em http://www.businessrulesforum.com/BOF_zachman.htm em 29/09/2002.

[ZIMB01] ZIMBRÃO, Geraldo et al., *ATENAS: um sistema gerenciador de regras de negócio*. XV Simpósio Brasileiro de Engenharia de Software, Instituto Militar de Engenharia, Rio de Janeiro-RJ, pág. 338-343, Out/2001.

ANEXO A - O SELIC NO SISTEMA FINANCEIRO NACIONAL

A.1 - Ambientação

Para uma boa compreensão do **SELIC** e de sua relevância para as atividades econômicas do país, inicialmente será necessária uma breve conceituação sobre as políticas econômicas, com atenção especial para a política monetária e as operações de mercado aberto, um de seus instrumentos. Após esta ambientação será detalhada a importância do **SELIC** como ferramenta operacional do mercado aberto para, em seguida, descrevê-lo como sistema de informação abordando os principais problemas existentes**.**

As políticas econômicas de um país referem-se às ações práticas desenvolvidas pelo governo com a finalidade de conduzir o sistema econômico para alcançar objetivos econômicos programados. No Brasil, as políticas econômicas seguem a orientação política da Presidência da República e técnica do Ministério da Fazenda. Neste contexto, os instrumentos de ação mais significativos e utilizados com mais freqüência são as políticas *fiscal*, *cambial* e *monetária*. [BCB96]

A política *fiscal* refere-se à administração das finanças públicas, considerando as receitas e despesas do governo. A política *cambial* refere-se às ações que determinam as relações entre a moeda nacional e as moedas estrangeiras. Finalmente, a política *monetária* refere-se às ações que controlam a oferta da moeda e do crédito para o sistema econômico, tendo no **SELIC** um importante componente operacional. [BCB95, BCB96]

A.2 - Instrumentos de Política Monetária

A política monetária controla a quantidade de moeda no mercado à disposição dos agentes econômicos, estimulando ou restringindo as atividades econômicas e influenciando os preços de bens e serviços. Seu principal objetivo portanto, é a manutenção do poder de compra da moeda nacional. Para concretizar esta tarefa, a política monetária se utiliza de três instrumentos: o *depósito compulsório*, o *redesconto* e o *mercado aberto* (*open market*). [BCB96]

O *depósito compulsório* representa uma parcela dos depósitos que os bancos comerciais devem recolher obrigatoriamente junto ao BC. Este por sua vez altera os percentuais de depósito compulsório conforme a necessidade de expansão ou retração dos meios de pagamento para adequar a quantidade de moeda em circulação aos objetivos da política monetária. [BCB96]

As operações de *redesconto* são empréstimos que o BC concede aos bancos comerciais com o intuito de auxiliá-los a superar dificuldades momentâneas de caixa que podem ocorrer devido à perda de depósitos ou ao não-recebimento do pagamento de empréstimos realizados. [BCB96]

O terceiro instrumento de política monetária, de maior importância para o escopo deste trabalho, são as operações de *mercado aberto*, que consistem na compra e venda de títulos públicos por parte do BC. As operações de mercado aberto apresentam uma vantagem sobre os outros instrumentos de política monetária porque sua direção pode ser alterada quase instantaneamente, bastando que o BC atue no mercado comprando ou vendendo títulos públicos. [BCB96, BCB97]

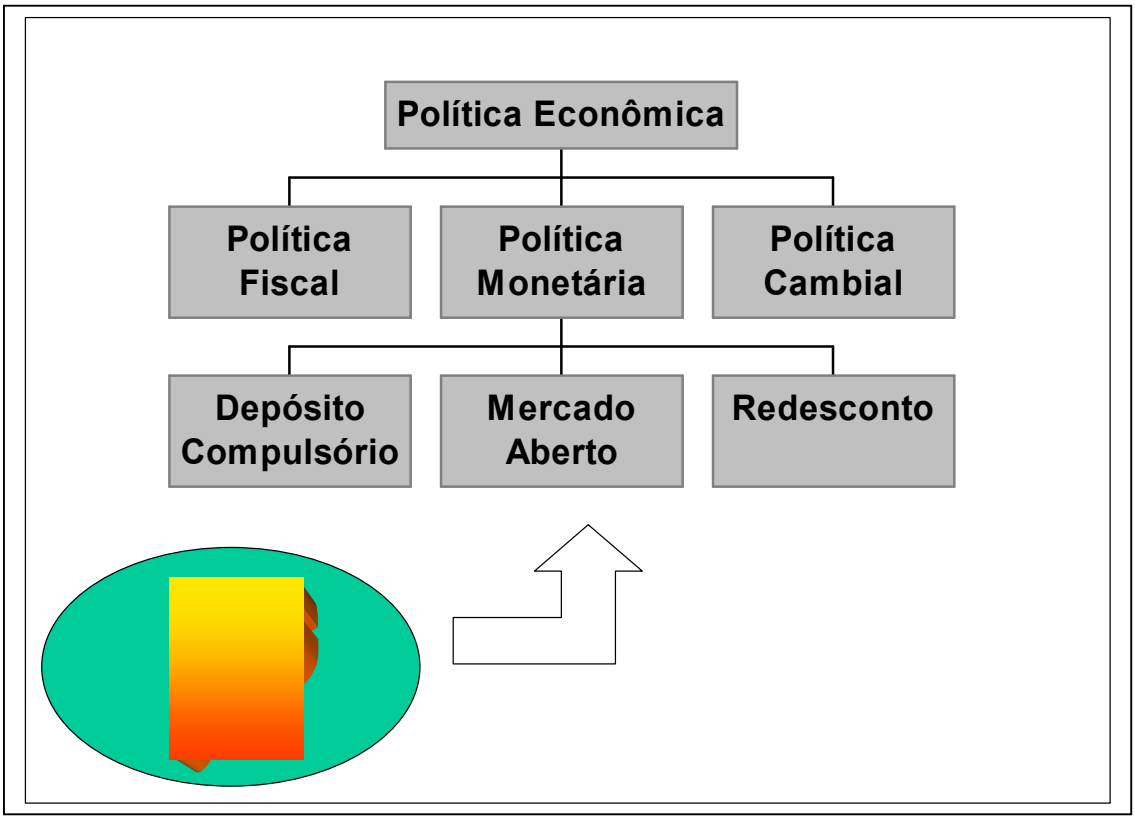

Figura A.1 – **Política econômica e atuação do SELIC na política monetária.**

Até aqui vimos que o BC exerce importante papel na execução da política econômica de um modo geral e na política monetária em particular. Portanto, antes de um maior aprofundamento sobre o **SELIC** e o mercado aberto, é interessante contextualizar a importância do BC no Sistema Financeiro Nacional (SFN).

A.3 – Sistema Financeiro Nacional (SFN)

As economias modernas, que possuem um elevado grau de complexidade no relacionamento entre os agentes econômicos, necessitam de um sistema que organize e facilite a circulação da moeda. Esse papel é desempenhado pelo sistema financeiro, que é responsável pela intermediação da moeda entre os agentes econômicos.[BCB97]

Sistema Financeiro é, portanto, o conjunto de instituições públicas e privadas que possibilita a transferência de recursos dos ofertadores ou poupadores para os tomadores ou investidores. [BCB97]

O SFN, por sua vez, apresenta uma subdivisão em dois subsistemas. O subsistema normativo é composto pelo Conselho Monetário Nacional (CMN), a Comissão de Valores Mobiliários (CVM), o Conselho de Recursos do Sistema Financeiro Nacional (CRSFN) e o BC. O subsistema operativo é composto pelas instituições financeiras, bolsas de valores, bancos governamentais, sociedades corretoras e sociedades distribuidoras de títulos e valores mobiliários. [BCB97]

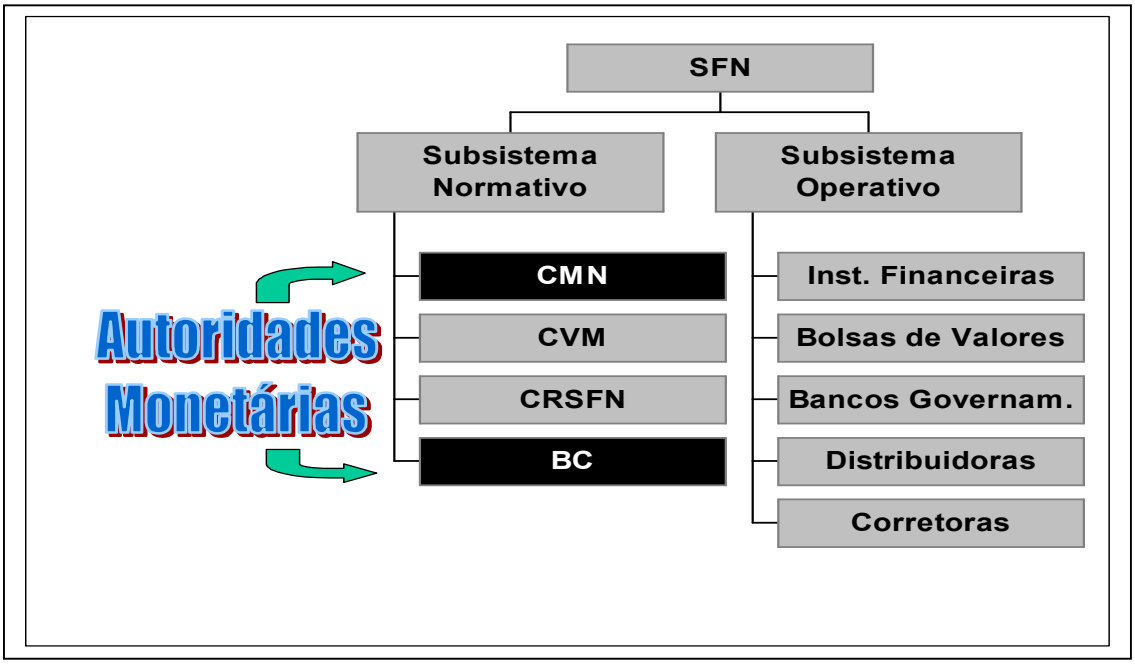

Figura A.2 – **Sistema Financeiro Nacional e identificação das Autoridades Monetárias.**

Dentro do subsistema normativo, encontramos as autoridades monetárias representadas no SFN pelo CMN e pelo BC. O CMN é o órgão de cúpula do SFN, cabendo-lhe a fixação de diretrizes gerais para as políticas monetária e creditícia a serem implementadas, abrangendo também as políticas fiscal, cambial, salarial e de alocação de recursos na economia. Resumidamente, suas principais funções são:

- Autorizar a emissão de papel-moeda, notas e moedas metálicas;
- ***Estabelecer diretrizes para as operações do BC com títulos públicos***;
- Regulamentar as operações de redesconto;
- Regulamentar as operações de câmbio e a política cambial;
- Determinar as características gerais das cédulas e das moedas;
- Aprovar os orçamentos monetários, preparados pelo BC, por meio dos quais se estimarão as necessidades de moeda e de crédito;

- Adaptar o volume dos meios de pagamento às reais necessidades da economia.

O BC, criado pela Lei 4.595, de 31.12.64, é o órgão executor da política monetária, controlando a oferta de moeda e de crédito, além de exercer a regulamentação e fiscalização de todas as atividades de intermediação financeira no país. Suas principais atribuições são:

- Emitir papel-moeda e moedas metálicas, nas condições e limites autorizados pelo CMN;
- Fixar os coeficientes de depósito compulsório sobre depósitos à vista e a prazo;
- Receber os depósitos compulsórios dos bancos comerciais;
- Realizar operações de redesconto e empréstimos a instituições financeiras;
- Conceder a autorização para o funcionamento das instituições financeiras, fiscalizá-las e aplicar as penalidades previstas, quando necessário;
- ***Efetuar, como instrumento de política monetária, operações de compra e venda de títulos públicos federais***[7]***;***
- Controlar o crédito e as taxas de juros;
- Administrar as reservas cambiais do país;
- Executar os serviços do meio circulante;
- Efetuar o controle de capitais estrangeiros e ser depositário dos recursos oficiais em ouro e moeda estrangeira.

---

[7] O BC pode aumentar ou diminuir o volume de dinheiro disponível para os bancos comerciais ao comprar ou vender títulos no mercado aberto, o qual tem sido um instrumento de muita eficácia para a política monetária, pois permite ao BC um melhor controle da oferta monetária, possibilitando a remuneração, a curto prazo, de recursos ociosos que estão à espera de sua real destinação.

Uma vez definidos a posição do BC no SFN e os conceitos de política monetária, assim como seus instrumentos, é possível entender melhor o funcionamento do mercado aberto.

A.4 – O Mercado Aberto

No início da década de 70, ainda sob o efeito do Milagre Brasileiro, o governo passou a estimular a captação maciça de recursos para investimento no país que passaram a ser obtidos junto aos agentes econômicos através da emissão de títulos públicos por parte do BC e do Tesouro Nacional. [BCB97]

A primeira emissão de títulos de qualquer espécie caracteriza o mercado *primário*, por ser a primeira tentativa de captação de recursos por parte do emissor. Cada título emitido possui uma data de vencimento específica[8] na qual o emissor se compromete a pagar o valor de face descrito mais um percentual de correção indicado por uma taxa de remuneração previamente estabelecida. São exemplos de títulos públicos as Notas do Tesouro Nacional (NTN), os Bônus do Banco Central (BBC), as Letras Financeiras do Tesouro (LFT) e as Letras do Tesouro Nacional (LTN). [BCB95, BCB97]

A existência de um mercado de financiamento desta natureza somente é possível se houver liquidez nos objetos negociados (neste caso, os papéis). Esta liquidez é a mola propulsora das negociações entre as instituições, o que caracteriza o mercado *secundário*. [BCB97]

O prazo de vigência dos papéis é determinado pela diferença entre suas datas de emissão e vencimento. Durante este período as instituições financeiras realizam entre si operações com os papéis tendo em vista cumprir seus compromissos de financiamento. Instituições que necessitam de dinheiro podem vender papéis para aquelas que desejam aplicar alguma sobra de recursos. As

[8] A combinação de um título com uma data de vencimento é conhecida no mercado através do jargão denominado *papel*.

operações com os papéis são diárias e na data de vencimento, a instituição detentora do papel recebe o valor de *resgate* junto ao órgão emissor. [BCB97]

A.5 – Um Sistema de Registro

Todas as operações envolvendo títulos públicos passaram a ser registradas junto ao BC num sistema de custódia que controlava as posições de cada instituição financeira. O departamento responsável por esta atividade no BC é o DEMAB (Departamento de Operações do Mercado Aberto), no qual se localiza a DICEL (Divisão de Administração do **SELIC**)[9], conforme extrato do organograma na Figura A.3. [BCB95, BCB98]

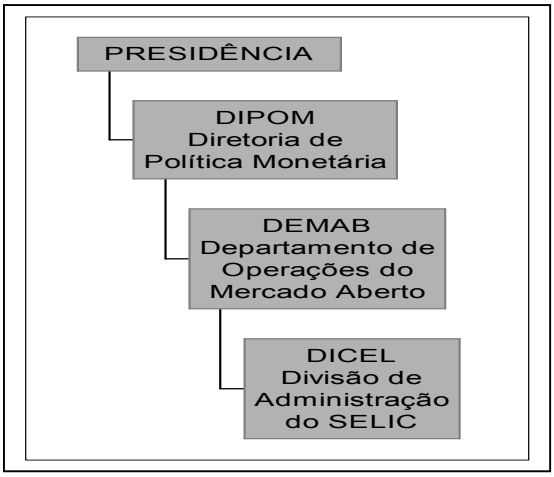

Figura A.3 **– Localização na estrutura do BC.**

[9] A sigla DICEL significa originariamente **DI**visão de **C**ustódia **E L**iquidação. Posteriormente, com a implantação do **SELIC,** a sigla permaneceu, mas sua descrição passou a ser Divisão de Administração do **SELIC.**

Com os escassos recursos computacionais do início da década de 70 o controle sobre as operações de mercado aberto era fundamentalmente manual, uma vez que os papéis existiam fisicamente e eram armazenados em compartimentos de segurança dentro do prédio do BC. Nesta época a custódia de títulos era física.

Com o desenvolvimento do mercado de títulos públicos[10] e a utilização crescente de recursos de informática nas instituições financeiras, as práticas manuais existentes no BC como órgão regulador ficaram insustentáveis e, o que era pior, injustificáveis, uma vez que soluções computacionais estavam mais acessíveis. As operações de mercado aberto não poderiam prescindir de um sistema automatizado, havendo o risco de limitação da liquidez e conseqüentemente de perda de eficiência do SFN como um todo. [BCB97]

Através de um convênio estabelecido entre o DEMAB e a ANDIMA (Associação Nacional das Instituições do Mercado Aberto) em 21.02.1974, foram viabilizados recursos para aquisição de equipamento e suprimentos necessários, além da contratação de mão-de-obra especializada, para suporte e desenvolvimento de aplicativos. O DEMAB, representando o BC, entraria com os conhecedores do negócio (as pessoas que conceberam o **SELIC**) e com as instalações físicas, já que o equipamento seria instalado no prédio do BC na cidade do Rio de Janeiro. [BCB95]

Nasce então o **SELIC**, um sistema de teleprocessamento eletrônico que se destina ao **registro** de títulos de emissão do Tesouro Nacional, BC, estados e municípios, e de depósitos interfinanceiros em contas abertas em nome dos participantes, processando as operações de movimentação geral, bem como as rotinas de pagamento de juros, resgates e ofertas públicas, promovendo a conseqüente **liquidação financeira** nas contas de reservas bancárias das instituições envolvidas. [BCB95, BCB98]

---

[10] Especialmente o crescimento da troca de reservas entre instituições que negociavam LTNs.

ANEXO B – PADRÃO SELIC DE PROGRAMAÇÃO

B.1 - Estrutura geral do programa COBOL

- O programa COBOL deverá ser dividido em seções (SECTIONS).
- Os nomes das seções deverão ser iniciados por um verbo no infinitivo.
- Os nomes dos parágrafos de uma seção deverão ser iniciados pelas iniciais da seção.
- Toda SECTION deverá ser iniciada por um parágrafo *///*-INICIO e terminar por um parágrafo *///*-FIM, onde *///* são as iniciais da SECTION.

Exemplo:

```
      .
      .
      .
PROCEDURE DIVISION.
PROCESSAR-APURACAO-LEILAO SECTION.
PAL-INICIO.
      <.......>
      PERFORM OBTER-AMBIENTE-EXECUCAO.
      PERFORM OBTER-DESIGNATOR-AGENDAS.

      <.......>
PAL-FIM.
     STOP RUN.

OBTER-AMBIENTE-EXECUCAO SECTION.
OAE-INICIO.
      <.........>
      <.........>
OAE-FIM
```

B.2 - Implementação das estruturas básicas

Pseudocódigo	COBOL
SE <condição> ENTÃO	IF <condição>
<.....>	<.....>
<.....>	<.....>
SENÃO	ELSE
<.....>	<.....>
<.....>	<.....> .
FIM SE	
PARA *i* DE *ini* ATÉ *fim*	MOVE *ini* TO *i* .
<.....>	LOOP.
<.....>	IF *i* > *fim*
<.....>	GO TO FIM-LOOP .
<.....>	<.....> .
FIM PARA	<.....> .
	<.....> .
	<.....> .
	ADD 1 TO *i* .
	FIM-LOOP .
ENQUANTO <condição>	LOOP .
<.....>	IF NOT <condição>
<.....>	GO TO FIM-LOOP .
<.....>	<.....> .
<.....>	<.....> .
FIM ENQUANTO	<.....> .
	<.....> .
	FIM-LOOP .

```
REPITA
    <.....>
    <.....>
    <.....>
    <.....>
ATÉ <condição>
```

```
LOOP
    CORPO
SE <condição_1>   TF_COND_1
            ...
        SE <condição_n>   TF_COND_n
<.......>
        <.......>
<.......>
<.......>
    FIM CORPO
    TRATAMENTO FINAL
        TF_COND_1:
            <.......>
            <.......>

TF_COND_n:
            <.......>
    <.......>

    FIM TRATAMENTO FINAL
FIM LOOP
```

```
LOOP .
        <.......> .
        <.......> .
        <.......> .
        <.......> .
        IF NOT <condição>
            GO TO LOOP .
FIM-LOOP .
```

```
        LOOP.
            IF <condição_1>
          GO TO TF-1 .
            ...
            IF <condição_n>
     GO TO TF_N .

<.......> .
<.......> .
            <.......> .
            <.......> .
        GO TO LOOP .
        *Tratamento final
         TF-1.
            <.......> .
<.......> .
            GO TO FIM-LOOP .

 TF-N .
            <.......> .
        <.......> .
         FIM-LOOP .
```

```
REGRA                                          IF <condição 1>
    SE <condição 1>  CP_1              *           Caso particular 1
        ...                                            <.......> .
    SE <condição n>  CP_N                          <.......> .
                                                       GO  TO  FIM-REGRA  .

    CASO GERAL
        <.......>                                IF <condição n>
        <.......>                              *       Caso particular n
        <.......>                                      <.......> .
        <.......>                                      <.......> .
    FIM CASO GERAL                                     GO  TO  FIM-REGRA  .

    CASOS PARTICULARES
      CP_1:                                    *       Caso geral
          <.......>                                    <.......> .
          <.......>                                    <.......> .
                                                       <.......> .
       CP_N:                                   <.......> .
           <.......>
           <.......>                            FIM-REGRA .

    FIM CASOS PARTICULARES
FIM REGRA
```

B.3 - Regras gerais de programação

- NUNCA usar:

  PERFORM *nome_de_parágrafo*

  PERFORM *nome_de_seção* VARYING ... ou
  UNTIL ....

  GO TO *nome_de_parágrafo,* exceto para implementar uma das estruturas definidas no item 2.

- Os comentários devem acompanhar a estrutura a que pertencem. Exemplo:

Correto

```
LOOP.
   IF <condição>
                GO TO FIM-LOOP.
    *       Início do loop
            <......> .
            <......> .
            <......> .
   IF <cond_1>
    *           Condição 1 atendida
                <.........>
                <.........>
           ELSE
    *           Condição 1 NÂO atendida
                <.........>
                <.........>
                <.........> .

           <......> .
           <......> .
           <......> .
           GO TO LOOP.
     FIM-LOOP.
```

Incorreto

```
LOOP.
IF <condição>
                GO TO FIM-LOOP.
*Início do loop
            <......> .
            <......> .
            <......> .
IF <cond_1>
*Condição 1 atendida
                <.........>
                <.........>
           ELSE
*Condição 1 NÂO atendida
                <.........>
                <.........>
                <.........> .

           <......> .
           <......> .
           <......> .
           GO TO LOOP.
     FIM-LOOP.
```

As SECTIONS devem ter função única e bem definida, e possuir um nome que as identifique claramente. Por exemplo, a seção

MONT-TELA-CRIT-SEL SECTION.

deve ser dividida em duas:

MONTAR-TELA-LEILOES SECTION. e

CRITICAR-SELECAO-LEILAO SECTION.

Cada SECTION deve ocupar preferencialmente uma página, eventualmente duas e no máximo três páginas.

B.4 Variáveis

Deve-se evitar o uso indiscriminado de variáveis globais.
Os nomes das variáveis devem identificar claramente seu conteúdo e devem começar pelas iniciais do nome da seção onde são declaradas.
As variáveis devem ser declaradas separadamente para cada SECTION.

Exemplo:

WORKING-STORAGE SECTION.

...

* Variáveis usadas no modulo CALCULAR-VALOR-FINANCEIRO

```
01  CVF-FINANCEIRO                    PIC 9(9)V99.

*   Variáveis usadas no modulo APURAR-LEILAO
01  AL-VOLUME-PREVIA-FORMATADO        PIC ZZZZZZ.ZZZ.ZZZ.ZZZ.
01  AL-TXCOT-PREVIA-FORMATADO         PIC ZZZ,ZZZZ.
```

ANEXO C – ESQUEMA DAS PADRONIZAÇÕES PARA IDENTIFICAÇÃO DE TERMOS DE NEGÓCIO

Na Figura C.1 observamos uma parte do modelo utilizado para parametrização de padrões para identificação dos termos de negócio. As tabelas acinzentadas representam as entidades já existentes no **Source Inspector** .

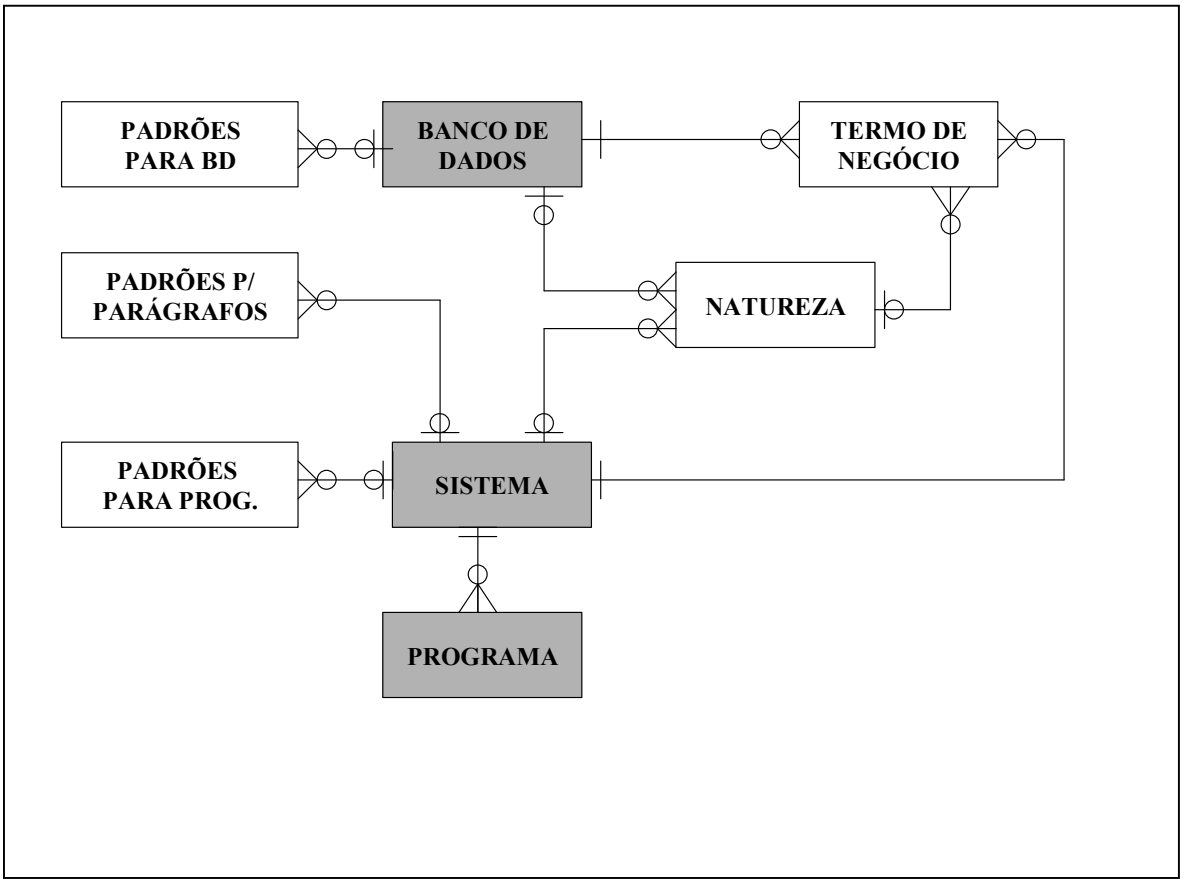

Figura C.1 – **Modelo simplificado contendo as tabelas de Natureza, Termos de Negócio e Padrões.**

C.1 - Termo de Negócio

Atributo	Descrição
Termo de Negócio	Referência ao termo de negócio dentro do fonte
Descrição Sucinta	Descrição do termo de negócio com o menor número possível de palavras
Descrição Ampliada	Descrição mais detalhada (dicionarização)
Origem	(Extração via SI ou informado pelo especialista de domínio)
Nome Original	Se vier de extração
Data de Registro	Data do Cadastramento ou do resultado da extração
Abrangência	Termo de Negócio específico de um programa, de um sistema ou do negócio como um todo (global)

Quadro C.1 – **Descrição da tabela de Termos de Negócio.**

C.2 - Natureza

Atributo	Descrição
Código	Código abreviado da Natureza
Descrição Sucinta	Descrição da Natureza do Atributo
Descrição Ampliada	Dicionarização da Natureza do Atributo
Posição	No termo de negócio, onde a seqüência de caracteres se encontra (sufixo, prefixo ou meio)
Caracter Separador	Indica a existência de algum caracter separador (como o caracter "-", por exemplo)
Aplicável	Indicador que habilitará a avaliação da natureza no processo

	de extração.

Quadro C.2 – **Descrição da tabela de Naturezas de atributo.**

No quadro a seguir vemos um exemplo de algumas possíveis ocorrências desta tabela de naturezas:

Cod	Descrição	Descrição Ampliada
ALQ	ALÍQUOTA	Percentual com que determinado tributo incide sobre o valor da coisa tributada
CAT	CATEGORIA	Classificação que expressa a hierarquia em uma determinada classe.
CLA	CLASSE	Num determinado contexto, grupo ou divisão que apresenta características semelhantes.
COD	CÓDIGO	Caracter(es) reunido(s) de acordo com regras que permitem a representação de dados de uma forma biunívoca e discreta.
CTE	CONSTANTE	Caracter(es) numérico(s) ou não que não sofrem modificação.
COT	COTAÇÃO	Determinação dos preços pelo qual se negociam mercadorias, títulos, ações de bancos fundos públicos, moedas estrangeiras, nas bolsas ou nas praças de comércio.
DAT	DATA	No registro de um fato, data que é escolhida como referência do tempo em que as grandezas pertinentes ao fato são medidas. ***Obs***.: A data deve ser composta por dia, mês e ano, independente se em padrão internacional (aaaa-mm-dd), americano (mm/dd/aaaa) ou europeu (dd.mm.aaaa).
DES	DESCRIÇÃO	Representação descritiva pormenorizada de um objeto.
DIG	DÍGITO	Caracter para identificação posicional ou controle de determinado objeto.
END	ENDEREÇO	Caracterização de um lugar.
FAT	FATOR	Coeficiente que, submetido à operação de produto, define o valor de um novo dado.
HOR	HORA	No registro de um fato, hora que é escolhida como referência do tempo em que as grandezas pertinentes ao fato são medidas. ***Formação***: hh ou hh:mm ou hh:mm:ss .
IDE	IDENTIFICADOR	Conjunto de caracteres que identificam um objeto, reunidos sem obedecer a regras.
IND	INDICADOR	Dado indicativo de um estado ou uma situação.
IDC	ÍNDICE	Número que indica a variação ocorrida em uma grandeza ou a relação entre os valores de qualquer medida.
MAT	MATRÍCULA	Inscrição de uma pessoa física em registros oficiais ou particulares com o fim de legalizar o exercício de certas profissões ou autorizar o gozo de certos direitos.
MED	MÉDIA	Numa distribuição, valor que se determina segundo uma regra estabelecida a priori e que se utiliza para representar todos os valores da distribuição.
NOM	NOME	Título, qualificação. Palavra que designa pessoa, coisa, animal, entidade.
NUM	NÚMERO	Identificador numérico.
PER	PERÍODO	Tempo transcorrido entre duas datas ou dois fatos.
PRE	PREÇO	Valor unitário de uma coisa vendida ou comprada, expresso em dinheiro.
SIG	SIGLA	Abreviatura. Reunião das letras iniciais dos vocábulos fundamentais de uma denominação ou título.
QTD	QUANTIDADE	Número de unidades, ou medida, que determina um conjunto de coisas consideradas como equivalentes e suscetíveis de acréscimo ou diminuição.
TIP	TIPO	Característica associada a um elemento ou conjunto de elementos, que permite realizar uma classificação de objetos pertencentes a classes diferentes.
UNI	UNIDADE	Qualquer unidade de medida. Quantidade que se toma arbitrariamente para termo de comparação entre grandezas da mesma espécie.
VAL	VALOR	Número que expressa o valor de algo em uma unidade.

Quadro C.3 – **Exemplos de Natureza.**

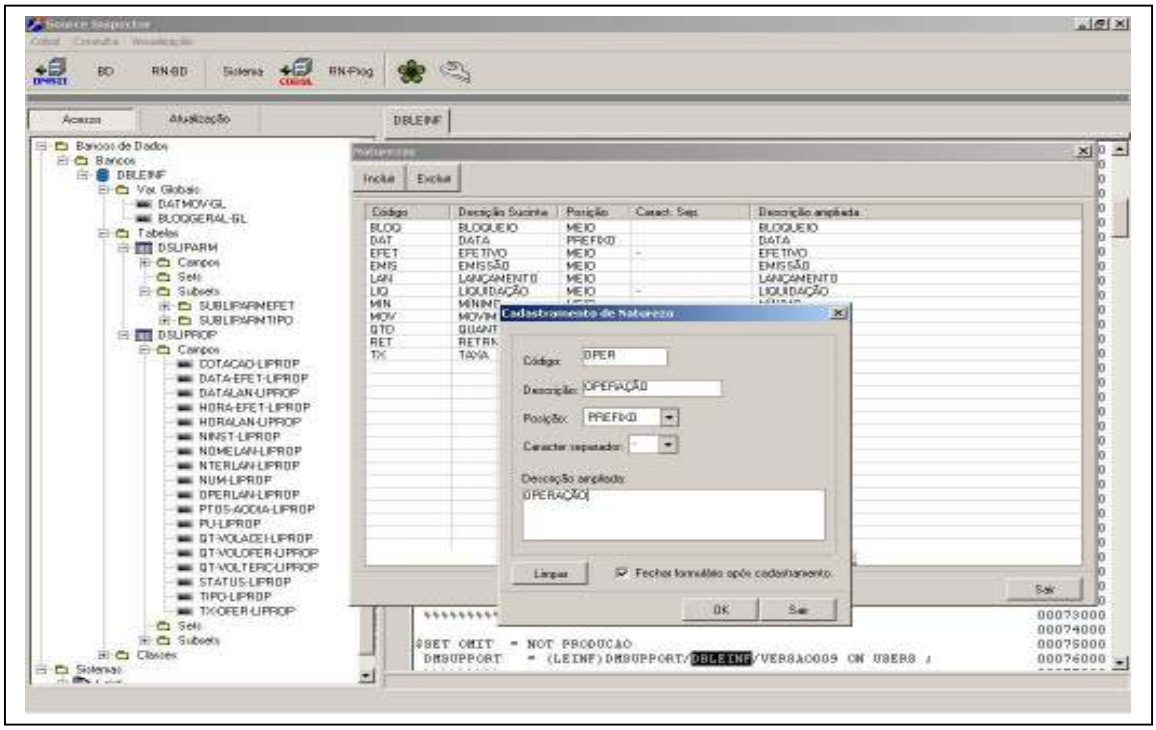

Figura C.2 – **Interface do** Source Inspector **para cadastramento de naturezas. A esquerda da tela temos a árvore com os itens de dado do Dasdl e ao fundo o código fonte do arquivo descritor do banco de dados.**

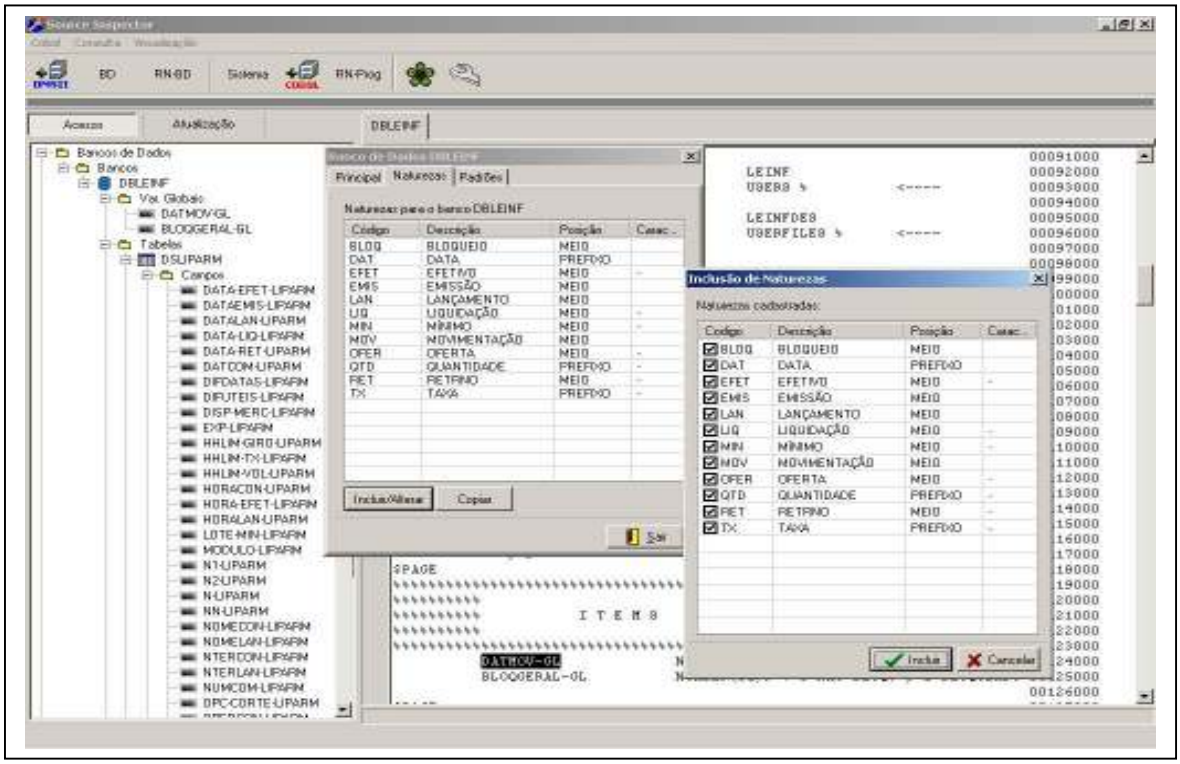

Figura C.3 – **Cadastramento de naturezas para o banco DBLEINF. As naturezas são selecionadas pelo próprio especialista de domínio.**

C.3 - Padrões de BD

Atributo	Descrição
Tipo de Padrão	Estabelece sobre qual componente o padrão será aplicado. Este componente pode ser um DATASET, um SET, um SUBSET ou um Campo;
Posição de filtragem	Parte do item de dado que será ignorada para obter o termo (prefixo, sufixo, meio)
Caracter Separador	Indica a existência de algum caracter separador (como o caracter "-", por exemplo)
Número de Caracteres	Quando não houver caracter separador, número de caracteres a ser ignorado (vale apenas para prefixo e sufixo)
seqüência	seqüência determinada de caracteres para ser identificada e retirada do item de dado, deixando apenas o termo de negócio.

Quadro C.4 – **Descrição da tabela de Padronização.**

Abaixo alguns exemplos de como os padrões de BD podem ser utilizados:

Tipo de Padrão	Pos. Filtragem	Carac. Separador	Núm Caracteres	seqüência
DATASET	PREFIXO		4	
SET	PREFIXO		(desnecessário)	SET
SUBSET	PREFIXO		(desnecessário)	SUB
Campo	SUFIXO	-		

Quadro C.5 – **Padrões cadastrados para um determinado banco de dados.**

Seguindo a padronização de BD acima, o SI produziria o seguinte resultado na extração de termos de negócio:

Componente	Nome Original	Termo de Negócio
DATASET	FINAPESSOAL	PESSOAL
DATASET	FINABANCO	BANCO
SET	SETNOMEBANCO	NOMEBANCO
SUBSET	SUBREGIAO	REGIAO
Campo	AGENCIA_FINABANCO	AGENCIA

Quadro C.6 – **Resultado da padronização sobre a extração de termos de negócio em bancos de dados.**

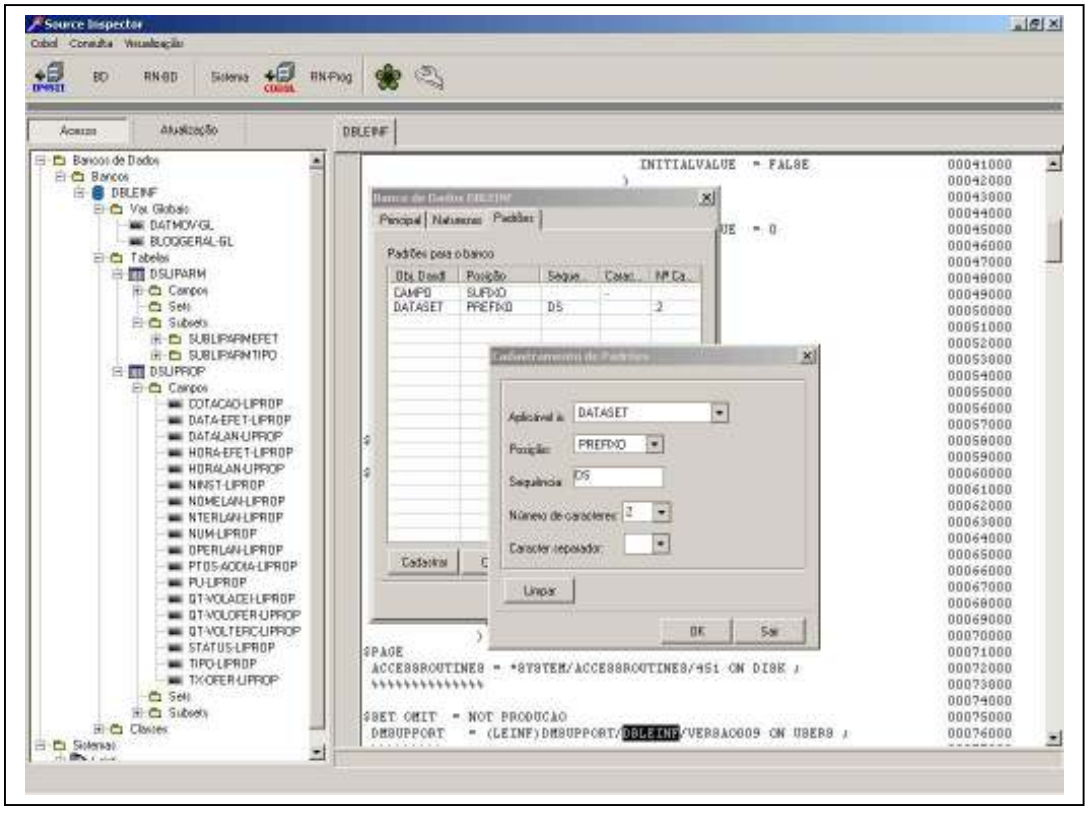

Figura C.4 – **Cadastramento de padrões para banco de dados.**

C.4 - Padrões para programas

Atributo	Descrição
Tipo de Padrão	Estabelece sobre qual componente o padrão será aplicado. Estes componentes pode ser: - variáveis de trabalho genéricas (WORKING-STORAGE SECTION); - tabelas do COBOL (itens de dados contendo cláusula OCURS); - cabeçalhos e linhas de detalhe em relatórios; - campos de arquivos; - variáveis de controle de iteração; - variáveis acumuladoras; - variáveis totalizadoras; - variáveis contadoras.
Posição de filtragem	Parte do item de dado que será ignorada para obter o termo (prefixo, sufixo, meio)
Caracter Separador	Indica a existência de algum caracter separador (como o caracter "-", por exemplo)
Número de Caracteres	Número de caracteres a ser ignorado quando a opção de posição de filtragem for prefixo ou sufixo e não houver caracter separador. Quando a opção de posição de filtragem for "meio", deve vir combinado com o item "seqüência" logo abaixo, a fim de caracterizar um tipo específico de variável.
Domínio do Negócio	Indica se o padrão será utilizado para identificar ou desprezar o item de dado como termo de negócio.
seqüência	seqüência determinada de caracteres para ser identificada e retirada do item de dado, deixando apenas o termo de negócio.

Quadro C.7 – **Descrição da tabela de Padronização para fontes COBOL.**

Tipo de Padrão	Pos. Filtragem	Carac. Separador	Núm Caracteres	seqüência	Domínio Negócio
Work	PREFIXO	-	(desnecessário)	W	N
Work	PREFIXO	-	(desnecessário)	WS	N
Tab	PREFIXO	-		TAB	S
Arq	PREFIXO	(não há)	3		S
Campo	SUFIXO	-	(desnecessário)		S
Cab	PREFIXO	(não há)	(desnecessário)	CAB	N
Detalhe	PREFIXO	(não há)	(desnecessário)	DET	N
VarControle	MEIO	(não há)	1	I	N
VarControle	MEIO	(não há)	1	J	N
VarControle	MEIO	(não há)	1	K	N
VarAcum	PREFIXO	(não há)	(desnecessário)	ACM	S
VarAcum	PREFIXO	(não há)	(desnecessário)	ACUM	S
VarAcum	PREFIXO	(não há)	(desnecessário)	ACUMULA	S
VarContador	PREFIXO	(não há)	(desnecessário)	CONT	N
VarTot	PREFIXO	(não há)	(desnecessário)	TOT	S
VarTot	PREFIXO	(não há)	(desnecessário)	TOTAIS	S
VarTot	PREFIXO	-	(desnecessário)	TOT	S
VarTot	PREFIXO	-	(desnecessário)	T	S
VarTot	PREFIXO	-	(desnecessário)	TOTAIS	S

Quadro C.8 – **Padrões cadastrados para um determinado sistema.**

Seguindo a padronização acima, o SI produziria o seguinte resultado na extração de termos de negócio:

Componente	Nome Original	Termo de Negócio
Work	**WS**-ACUMULA	(desprezado)
Tab	**TAB**-ESTADOS-FEDERACAO	ESTADOS-FEDERACAO
Arq	**ARQ**LIMITE	LIMITE
Campo	VALORMINIMO-**LIMITE**	VALORMINIMO
Cab	**CAB**REL01	(desprezado)
Detalhe	**DET**01	(desprezado)

Quadro C.9 – **Resultado da padronização sobre a extração de termos de negócio em programas, levando em conta quais os termos que efetivamente devem ser considerados.**

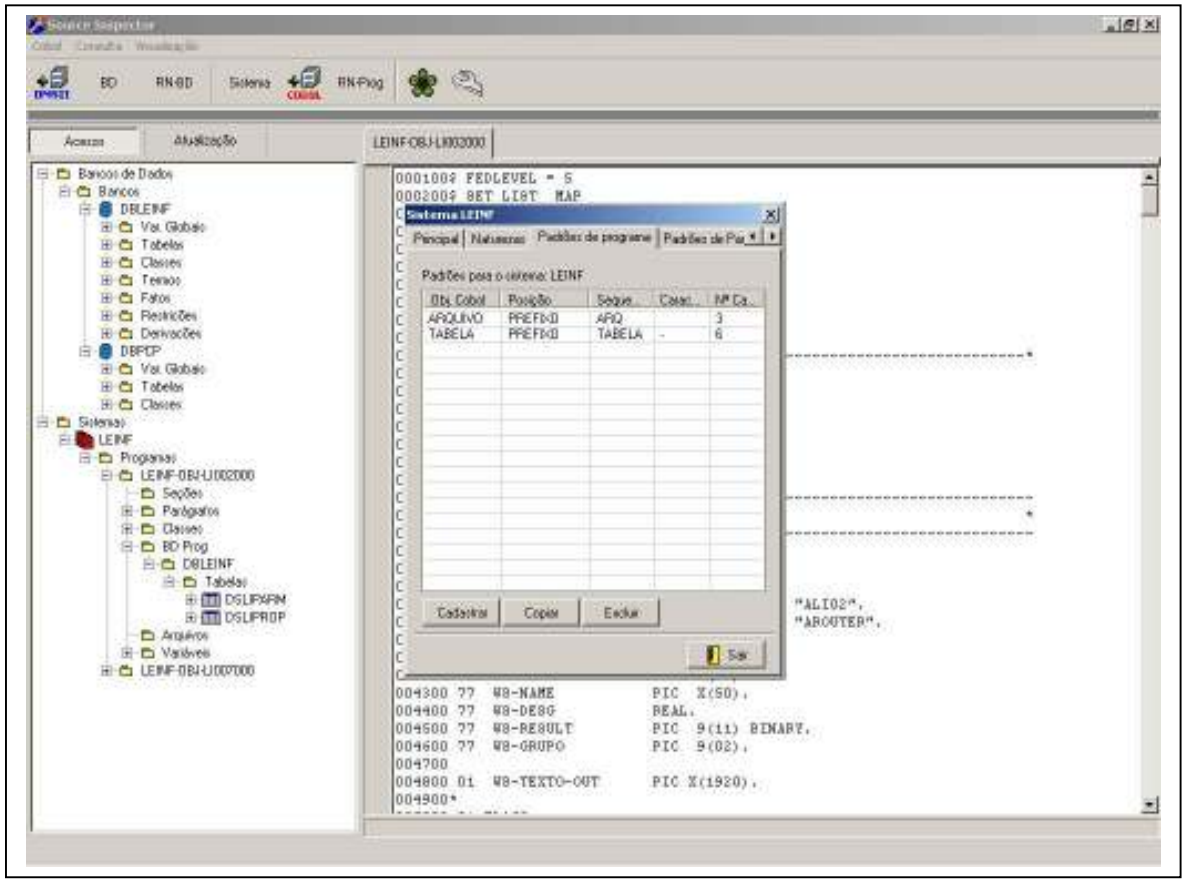

Figura C.5 – **Cadastramento de padrões para o programa LEINF-OBJ-LI002000.**

C.5 - Padrões de parágrafos

Atributo	Descrição
Tipo de Parágrafo	Estabelece o tipo de ação que determina cada parágrafo ou seção: - Leitura - Gravação - Abertura de Arquivos - Tratamento de Erros - Fim de bloco
seqüência	seqüência determinada de caracteres para ser identificada e caracterizar o parágrafo ou seção

Quadro C.10 – **Padronizações para os parágrafos**

Tipo de Parágrafo	Expressão
Leitura	LEITURA
Leitura	READ
Leitura	LER
Erro	ERRO
Erro	ERR
Gravação	GRAVA
Gravação	WRITE
Abertura	OPEN
Abertura	ABRIR
FimBloco	EXIT

Quadro C.11 – **Possíveis tipos de cadastramento.**

Seguindo a padronização acima, o Source Inspector produziria o seguinte resultado nas avaliações de seções e parágrafos:

Nome Original do Parágrafo	**Classificação**
LEITURA-REGISTRO	Parágrafo de Leitura
ABRIR-ARQUIVOS	Parágrafo de Abertura
TRATA-ERRO	Parágrafo de Erro
GRAVA-REG-SAIDA	Parágrafo de Gravação
OBTEM-DATA-**EXIT**	Parágrafo de Fim de Bloco

Quadro C.12 – **Resultado da padronização sobre a extração de nomes de parágrafos e seções.**

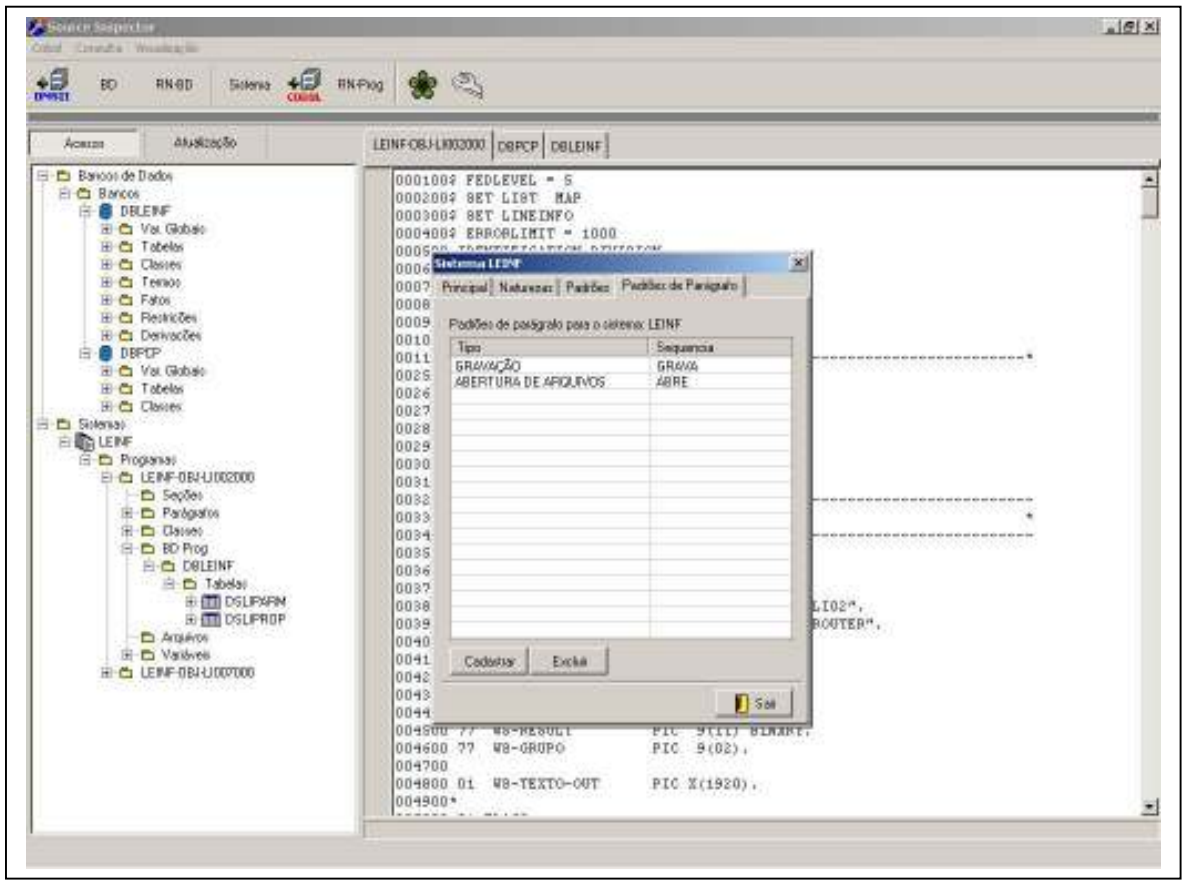

Figura C.6 – **Cadastramento de padrões de parágrafo para o programa LEINF-OBJ-LI002000.**

Exemplificando a utilização do layout, consideremos o banco de dados B1 apenas com o DATASET DS1, que, por sua vez, possua os campos c1, c2, c3 e c4. Este DATASET possui, por sua vez, dois sets: o S1 contendo os campos c1, c2 e c3, e o S2 com c1, c4 e c2. A geração do arquivo intermediário seria a seguinte:

Tipo de Registro	Conteúdo
0	B1
1	D1
2	c1 …
2	c2 …
2	c3 …
2	c4 …
3	S1 D1
4	c1
4	c2
4	c3
3	S2 D1
4	c1
4	c4
4	c2

Quadro C.13 – **Exemplo de arquivo intermediário de saída a partir de um fonte DASDL.**

Vale observar que o Tipo de Registro 2 (CAMPO) contém uma série de informações adicionais, representadas no quadro acima apenas pelas reticências após o nome do campo.

A.1.1.1 Item	Descrição		
NIVEL	Este item representa o nível de identação numa estrutura de grupo. Em fontes DASDL segue-se uma numeração seqüencial. Em fontes COBOL segue-se a numeração determinada pelo programa.		
FORMATO	**Nome**	**Cod**	**Descrição detalhada**
	Alpha	0	Alfanumérico
	Boolean	1	Booleano
	Inteiro	2	Inteiro
	Decimal	3	Decimal
	Group	4	Item de Grupo
	RType_pai	5	Record Type
	Ignora	9	Ignorados, ou seja, campos de um item de grupo com menos de três campos.
TAMANHO	Tamanho do campo		
OCORREDE	Número da ocorrência inicial de um vetor		
OCORRATE	Número da ocorrência final de um vetor		
NIVELCLASSE	Reconfiguração do nível original quando ignorado		
TIPOCLASSE	**Nome**	**Cod**	**Descrição detalhada**
	CLClasse	0	Classe
	CLAtrib	1	Atributo da Classe
	CLClasseAtrib	2	Item de grupo que se tornou uma classe
	CLIgnora	3	Correspondente ao item FORMATO/Ignora
NOME	Nome do Campo		

Quadro C.14 – **Detalhamento do tipo de registro 2 (CAMPO), semelhante tanto para os arquivos intermediários DASDL, quanto para os gerados a partir de fontes COBOL, como veremos a seguir.**

Exemplificando melhor o funcionamento do tipo de registro 2, agora num fonte COBOL, vamos supor o seguinte item de grupo:

```
01 Funcionario.
       05 Nome                    PIC X(40).
       05 Telefone.
             10 DDD               PIC 9(03).
             10 Numero            PIC 9(08).
       05 Endereco.
             10 Logradouro        PIC X(40).
```

Figura C.7 – **Exemplo de um item de grupo em fonte COBOL.**

O ModEx funciona recursivamente e, produzirá o seguinte quadro final:

Nível / Item	**CAMPO**			
	NIVEL	**NIVELCLASSE**	**TIPOCLASSE**	**FORMATO**
01 Funcionario	1	1	0	4
05 Nome	5	2	1	0
05 Telefone	5	2	3	9
10 DDD	10	2	1	2
10 Numero	10	2	1	2
05 Endereco	5	2	2	4
10 Logradouro	10	3	1	0
10 Numero	10	3	1	2
10 Complemento	10	3	1	0

Quadro C.15 – **Tipo de registro C2 após execução do ModTra.**

Para o tratamento desta estrutura, o ModTra executa sua rotina duas vezes. Na primeira, é montada a árvore relativa ao item de grupo, conforme detalha a Figura P.

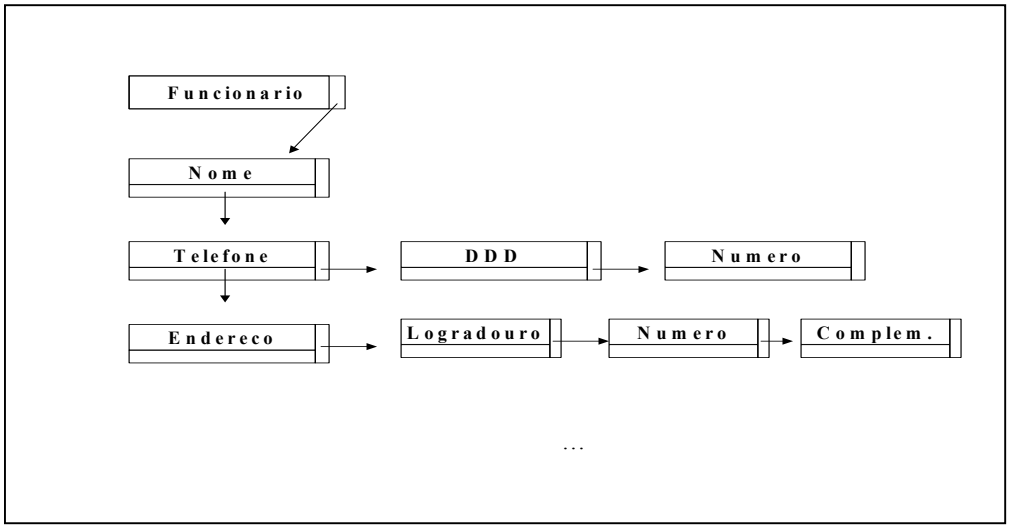

Figura C.8 **– Árvore montada para a estrutura FUNCIONARIO na primeira passada do ModTra.**

Em seguida, é feita uma segunda leitura que verifica o número de itens em cada grupo. Este é o processo recursivo em si, ou seja, a verificação de itens de grupo. À medida que a árvore vai sendo percorrida pela segunda vez, os valores vão sendo preenchidos até formar o quadro final.

ANEXO D - APURAÇÃO DIRETA NO SISTEMA OFPUB

D.1 - Relatórios do Sistema Oferta Pública Formal Eletrônica (OFPUB)

Antes de uma explicação mais detalhada sobre os relatórios, é importante ilustrar o fluxo de informações no processo do leilão de títulos públicos.

D.2 - Comunicado/Portaria

A oferta pública nasce quando o BC através de um comunicado ou o Tesouro Nacional através de uma portaria divulgam a intenção de colocar papéis para o mercado[11]. Num mesmo comunicado podem existir várias ofertas ou mesmo datas de liquidação distintas para cada papel ofertado.

Cada oferta de títulos federais é então cadastrada no OFPUB, onde são registradas, dentre outras informações, a data e a hora limite para o lançamento de propostas por parte das instituições. Confirmada a oferta, esta fica liberada para que as instituições cadastradas no SELIC lancem suas propostas no leilão, o que ocorre geralmente nos minutos finais que antecedem o prazo estabelecido pelo comunicado.

Findo o prazo para o lançamento de propostas, a mesa de operações do BC inicia o processo de apuração para algumas horas depois, divulgar o resultado para o mercado. Finalmente, conforme estabelecido no comunicado, ocorre a

---

[11] Isto ocorre no caso específico do OFPUB. No LEINF a ação do BC pode ser de comprador ou vendedor de papéis assim como tomador ou doador de dinheiro. Existe a intenção de sofisticar o sistema OFPUB a fim de que ele possa suportar um leilão de compra de papéis promovido pelo BC ou pelo Tesouro Nacional.

liquidação do papel via SELIC, quando a operação é financeiramente concretizada. Veja abaixo uma ilustração com a seqüência dos processos mais importantes da Oferta Pública.

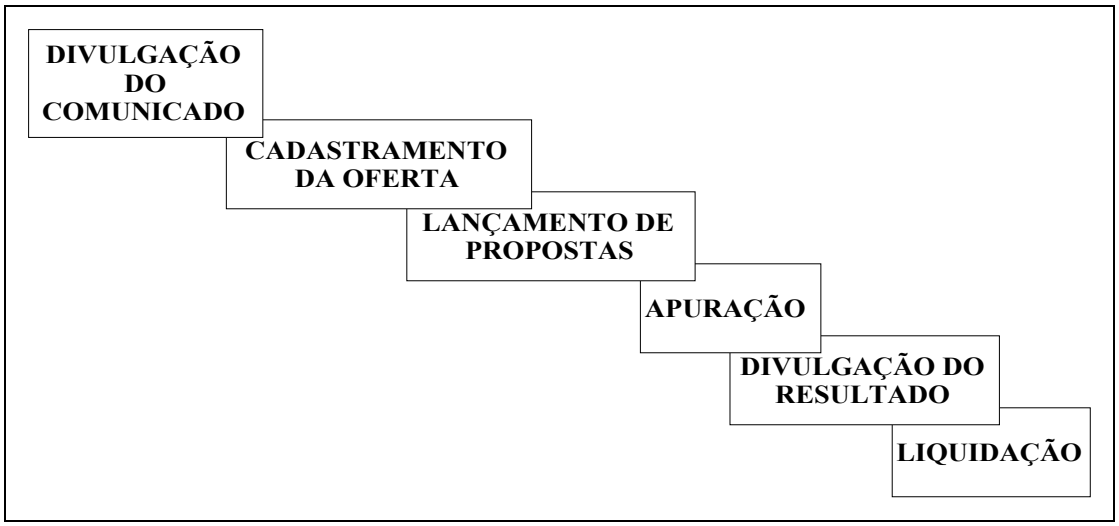

Figura D.1 – **Esquema simplificado das etapas do processo de apuração da Oferta Pública.**

D.3 - Apuração

Assim que o prazo para o leilão se encerra, a mesa de operações inicia o procedimento de apuração ***preliminar***. Esta apuração tem como objetivo validar o sucesso inicial do leilão através da verificação de itens como taxas média e mínima propostas, quantidade demandada e número de participantes.

Em seguida, após avaliações que levam em conta as diretrizes de política monetária e de controle da moeda, a chefia do DEMAB procede à aprovação do leilão. Caso este não contenha disparidades, é realizada uma apuração ***direta***. Em caso de taxas muito dispersas ou de quantidade demandada inferior à ofertada, o

BC pode optar por uma apuração com redistribuição, promovendo cortes de acordo com as taxas aceitáveis. Existe ainda a hipótese onde a autoridade monetária não considera o leilão válido e divulga o resultado como ***NDF (Nada Feito)***, o que na prática possui o mesmo efeito de um cancelamento da oferta[12].

Finalmente, existem leilões específicos onde o BC absorve a totalidade ou parte de uma oferta. Através do comunicado, o BC divulga o valor total da oferta de papéis. Durante o cadastramento da oferta no sistema OFPUB, num primeiro momento, o valor efetivamente oferecido ao mercado é cadastrado. Isto porque pode interessar ao BC manter em sua carteira uma parte do total da oferta. Neste caso, o BC aguardará o resultado do leilão para em seguida lançar a sua proposta com o PU (ou cotação) médio ponderado das propostas vencedoras, de forma a não alterar a apuração da oferta e o valor da taxa composta por estas propostas. Esta apuração é denominada ***TBC (Tudo Banco Central)***.

[12] Dificilmente uma oferta pública é cancelada pelo BC. Além dos problemas jurídicos em potencial que podem ocorrer caso alguma instituição financeira se sinta lesada, o cancelamento da oferta é um mau sinal para o mercado, demonstrando que a autoridade monetária não está administrando bem a colocação de papéis no mercado. Quando o resultado da oferta não é interessante para o BC, este apura o leilão na forma NDF (Nada Feito), não aceitando as propostas formuladas pelo mercado.

D.4 - Exemplo Prático de uma Apuração Preliminar:

Tomemos como exemplo uma oferta de 1.000.000, direta a 10%.

NUM	NOME INST	COT/PU	PROPOSTO	ACUMULADO	
1	ITAU	983,5	2.500	2.500	
2	BCN	983,5	2.500	5.000	
3	BOAVISTA	983	2.500	7.500	
4	BCN	982,5	2.000	9.500	
5	LLOYDS	982,5	2.500	12.000	
6	LLOYDS	981	5.500	17.500	
7	PATENTE	981	50.000	67.500	
8	BANDEIRANTES	981	50.000	117.500	
9	ITAU	980	2.000	119.500	
10	BANDEIRANTES	980	35.000	154.500	
11	PATENTE	980	35.000	189.500	
12	PATENTE	980	200.000	389.500	
13	BOAVISTA	980	200.000	589.500	
14	ITAU	965	5.000	594.500	
15	PATENTE	965	5.000	599.500	> LIM
16	LLOYDS	965	5.000	604.500	
17	BCN	965	5.000	609.500	
18	LLOYDS	940	8.000	617.500	
19	LLOYDS	850	1.500	619.000	
20	BANDEIRANTES	850	1.500	620.500	
21	ITAU	850	1.500	622.000	
22	BOAVISTA	850	65.000	687.000	> LIM
23	PATENTE	850	65.000	752.000	> LIM
24	ITAU	790	7.000	759.000	
25	BANDEIRANTES	790	70.000	829.000	

Quadro D.1 – **Resultado da apuração preliminar ordenando as propostas por PU. As colunas com observação (>LIM) indicam que a instituição ultrapassou o limite permitido para aquisição de papéis nesta oferta, conforme veremos mais detalhadamente a seguir.**

Além da ordenação natural das propostas, temos como resultado da apuração, um grupamento por faixa, conforme verificamos no quadro abaixo:

F A I X A			DIRETA a 10%	
QP	**PU**	**QUANT.**	**QP**	**QUANT.**
8	981	117.500	8	117.500
13	980	589.500	13	302.000
17	965	609.500	17	317.000
18	940	617.500	18	325.000
23	850	752.000	23	329.500
25	790	829.000	25	350.000

Quadro D.2 – **Grupamento por faixa de PU.**

Este resultado agrupa as propostas por PU de acordo com o percentual de apuração direta informado. No primeiro grupo, reúnem-se as melhores propostas até que seja atingido o número de instituições distintas estabelecido pelo percentual. Por exemplo: uma apuração direta a 10% determina que serão acumuladas as melhores propostas até que se tenhas dez instituições distintas. A 20% este número cairia para 5.

Estabelecido o PU do primeiro, todas as demais propostas com o mesmo PU também serão agrupadas nesta faixa. Os demais grupos são reunidos por PU. Quando o valor da oferta for atingido pelas quantidades acumuladas, é calculado o percentual do corte realizado na última proposta aceita. Exemplo: um corte de 30% numa proposta de 100.000 significa dizer que apenas 70.000 foram aceitos;

As demais colunas desta seção apresentam o seguinte significado: QP - Quantidade de Propostas agrupadas, acumuladas por faixa; PU ou Cotação a partir do qual foi feito o grupamento e finalmente QUANT - Quantidade acumulada;

D.5 Comentários sobre a apuração

Na primeira linha do grupamento por Faixa temos QP = 8 e PU = 981. Para obtermos a explicação deste número, basta acompanharmos a relação das propostas decrescentemente ordenada por PU. A primeira instituição relacionada é o ITAU. Em seguida temos BCN e BOAVISTA. A quarta melhor proposta é novamente do BCN, portanto o contador de instituições distintas permanece com o valor 3. Na quinta proposta surge uma instituição distinta, o LLOYDS, também autor da sexta proposta. Finalmente surge a proposta de número 7 de autoria do PATENTE, a quinta instituição distinta, que vai determinar o PU para a primeira linha do grupamento (qual seja 981). Porém, com este PU, existe uma outra proposta, a do banco BANDEIRANTE, que por esta razão também será considerada no mesmo grupo. Temos assim a explicação para QP = 8.

Para melhor compreensão da seção "DIRETA a 10%", devemos montar uma nova planilha, respeitando o limite de 100.000 papéis para cada instituição.

	QUANTIDADES POR INSTITUIÇÃO							
FAIXA PU	**ITAU**	**BCN**	**BOAVISTA**	**LLOYDS**	**PATENTE**	**BAND.**	**TOTAL**	**TOTAL ACUM**
981	2.500	4.500	2.500	8.000	50.000	50.000	**117.500**	**117.500**
980	2.000		**97.500**		**50.000**	35.000	**184.500**	**302.000**
965	5.000	5.000		5.000	**5.000**		**15.000**	**317.000**
940				8.000			**8.000**	**325.000**
850	1.500		**65.000**	1.500	**65.000**	1.500	**4.500**	**329.500**
790	7.000					**13.500**	**20.500**	**350.000**

Aceito[13]	**18.000**	**9.500**	**100.000**	**22.500**	**100.000**	**100.000**	**350.000**
Rejeitado			**167.500**		**255.000**	**56.500**	**479.000**

[13] Quando a proposta não for aceita, uma mensagem explicativa é impressa. Por exemplo, quando uma proposta não atinge o PU desejado pelo BC ou então não apresenta preço competitivo para ser incluída na relação das vencedoras, a mensagem TIPO INFERIOR é impressa.

Proposto	**18.000**	**9.500**	**267.500**	**22.500**	**355.000**	**156.500**	**829.000**

Quadro D.3 – **Resultado final da apuração direta por instituição.**

Observações:

- Linha 1 (PU = 981)
    - Quantidade BCN = 4.500. 2.500 da proposta 2 somados a 2.000 da proposta 4;
    - Quantidade LLOYDS = 8.000. 2.500 da proposta 5 somados a 5.500 da proposta 6.
- Linha 2 (PU = 980)
    - Quantidade BOAVISTA = 97.500. Houve corte de 102.500 da proposta 13, pois o limite total de 100.000 havia sido superado.
    - Quantidade PATENTE = 50.000; 35.000 da proposta 11 e 15.000 da proposta 12, onde 185.000 foram rejeitados por terem excedido o limite.
- Linha 3 (PU = 965)
    - Proposta 15 do PATENTE recusada, conforme assinalado pela mensagem ">LIM" no relatório. Por isso, QP para esta faixa nesta seção possui o valor de 16, e não 17 como está na primeira seção.

ANEXO E - DESCRIÇÃO DO BANCO DE DADOS DBOFPUB NO FORMATO DASDL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                                    %%%%%%%%%%
%%%%%%%%%%                 D B O F P U B                      %%%%%%%%%%
%%%%%%%%%%                                                    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


  UPDATE;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%           I T E N S   G L O B A I S                %%%%%%%%%%
%%%%%%%%%%                                                    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          DATMOV-GL             NUMBER (08)     %% FORMATO AAAAMMDD
         ;SLIBER-GL             NUMBER (01)     %% 0 = LIB, 1 = NAO LIB
         ;QTPROP-GL             NUMBER (03)     %% LIMITE PROP./INST.
         ;PRCDIR-GL             NUMBER (03)     %% % DA INST. DIRETA
         ;PRCRED-GL             NUMBER (03)     %% % DA INST. REDISTR.
         ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                                    %%%%%%%%%%
%%%%%%%%%%          D A T A   S E T   D S O F E R             %%%%%%%%%%
%%%%%%%%%%                                                    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                     OFERTAS PUBLICAS                         %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  DSOFER          STANDARD DATA SET
        (
          STATUS-OFER           NUMBER (01) % 0 = OK
                                            % 1 = ERRO
                                            % 2 = ANTERIOR
         ;HIST-OFER             NUMBER (01) % 0 = OFERTA NORMAL
                        INITIALVALUE 0      % 1 = OFERTA HISTORICO
         ;SIT-OFER              NUMBER (01) % 0 = NAO APURADO
                                            % 1 = DIRETA
                                            % 2 = REDISTR.
                                            % 3 = TBC
                                            % 4 = NDF
                                            % 5 = SINGLE
                                            % 6 = PRELIMINAR
                                            % 9 = NAO CONFIRMADO
         ;PAPEL-OFER            NUMBER (02) % ORDEM DO REG. NO DATA SET
         ;MODULO-OFER           NUMBER (06) % QTD. MODULAR MIN./PROPOSTA
         ;NUMCOM-OFER           NUMBER (04) % NUMERO DO COMUNICADO
         ;DATCOM-OFER           NUMBER (08) % DATA DO COMUNICADO
         ;DATLAN-OFER           NUMBER (08) % DATA DO LANCAMENTO
         ;DATLIQ-OFER           NUMBER (08) % DATA DA LIQUIDACAO
         ;TITULO-OFER           NUMBER (06) % CODIGO DO TITULO NO SELIC
         ;VENCTO-OFER           NUMBER (08) % DATA VENCIMENTO DO TITULO
         ;DATEMIS-OFER          NUMBER (08) % DATA EMISSAO DO TITULO
         ;TIPO-OFER             NUMBER (01) % 0 = NORMAL
                                            % 1 = CARTEIRA BC
                                            % 2 = SINGLE - NORMAL
                                            % 3 = SINGLE - CARTEIRA
         ;PRCDIR-OFER           NUMBER (03) % PERCENTUAL DIRETA
        ) ;

  SETOFERPAPEL  SUBSET OF DSOFER
  %%%%%%%%%%%%
        WHERE  (STATUS-OFER EQL 0 AND
                HIST-OFER   EQL 0)
```

```
       KEY                                                                  00303000
               ( TITULO-OFER                                                00304000
                ,VENCTO-OFER                                                00305000
                ,DATLIQ-OFER                                                00306000
                ,NUMCOM-OFER                                                00307000
                ,DATLAN-OFER                                                00308000
               )                                                            00309000
        ,INDEX SEQUENTIAL                                                   00310000
        ,NO DUPLICATES KEYCHANGEOK                                          00311000
       ;                                                                    00312000
  SETOFERCOD    SUBSET OF DSOFER                                            00313000
  %%%%%%%%%%                                                              00314000
       WHERE  (STATUS-OFER EQL 0 AND                                        00315000
               HIST-OFER   EQL 0)                                           00316000
       KEY     IS PAPEL-OFER                                               00317000
        ,INDEX SEQUENTIAL                                                   00318000
        ,NO DUPLICATES KEYCHANGEOK                                          00319000
       ;                                                                    00320000
  SETOFERHIST    SUBSET OF DSOFER                                          00321000
  %%%%%%%%%%                                                              00322000
       WHERE  (STATUS-OFER EQL 0 AND                                        00323000
               HIST-OFER   EQL 1)                                           00324000
       KEY    (TITULO-OFER                                                 00325000
              ,VENCTO-OFER                                                  00326000
              ,DATLIQ-OFER DESCENDING                                       00327000
              ,NUMCOM-OFER DESCENDING                                       00328000
              ,DATLAN-OFER DESCENDING                                       00329000
              )                                                             00330000
        ,INDEX SEQUENTIAL                                                   00331000
        ,NO DUPLICATES KEYCHANGEOK                                          00332000
       ;                                                                    00333000
$PAGE                                                                       00334000
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%00335000
%%%%%%%%%%                                                    %%%%%%%%%%00336000
%%%%%%%%%%          D A T A   S E T   D S P R O P             %%%%%%%%%%00337000
%%%%%%%%%%                                                    %%%%%%%%%%00338000
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%00339000
%%%%%                        PROPOSTAS                              %%%%%00340000
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%00341000
                                                                            00342000
 DSPROP         STANDARD DATA SET                                           00343000
 %%%%                                                                       00344000
       (                                                                    00345000
         STATUS-PROP           NUMBER (01)  % 0 = OK                      00346000
                                            % 1 = ERRO                    00347000
                                            % 2 = ALTERADO                00348000
        ;E-PROP                NUMBER (01)  % ESTADO DA PROPOSTA: 0 A 5   00349000
                                            %  0 - ACEITA                 00350000
                                            %  1 - TIPO INF. OU CRIT BC   00351000
                                            %  2 - ACIMA LIM. %           00352000
                                            %  3 - ACIMA LIM. PROP.       00353000
                                            %  4 - DECISAO BC             00354000
                                            %  5 - NAO CONFIRMADA         00355000
        ;CODINS-PROP           NUMBER (09)  % CODIGO SELIC DA             00356000
                                            % INSTITUICAO PROPONENTE      00357000
        ;NUMPRO-PROP           NUMBER (06)  % NUMERO DA PROPOSTA QUE,     00358000
                                            %  SE VENCEDORA, SERA A DO    00359000
                                            %  SELIC                      00360000
        ;PAPEL-PROP            NUMBER (02)  % ORDEM DO TITULO E DO VEN-   00361000
                                            %  CIMENTO NO DSOFER          00362000
        ;QUANTP-PROP           NUMBER (12)  % QUANTIDADE PROPOSTA         00363000
        ;QUANTC-PROP           NUMBER (12)  % QUANTIDADE CONSIDERADA      00364000
        ;QUANTA-PROP           NUMBER (12)  % QUANTIDADE ACEITA           00365000
        ;PU-PROP               NUMBER (12)  % PRECO UNITARIO ACEITO       00366000
        ;COT-PROP              NUMBER (12)  % COTACAO ACEITA              00367000
        ;PUPROPOSTO-PROP       NUMBER (12)  % PRECO UNITARIO PROPOSTO     00368000
        ;COTPROPOSTO-PROP      NUMBER (12)  % COTACAO PROPOSTA            00369000
        ;DATALAN-PROP          NUMBER (08)                                00375000
        ;FLAGALT-PROP          NUMBER (01)  % 0 = OK                      00376000
                                            % 1 = ALTERADO                00377000
        ;FLAGBC-PROP           NUMBER (01) ;% 0 = LANC. PELO MERCADO      00378000
```

```
                                        % 1 = LANC. PELO BACEN      00379000
      ) ;                                                           00380000
                                                                    00381000
SETPROPNUM       SUBSET OF DSPROP                                   00382000
%%%%%%%%%%                                                          00383000
        WHERE STATUS-PROP EQL 0                                     00384000
        KEY   NUMPRO-PROP                                           00385000
       ,INDEX SEQUENTIAL                                            00386000
       ,NO DUPLICATES                                               00387000
      ;                                                             00388000
                                                                    00389000
SETPROPPAPEL     SUBSET OF DSPROP                                   00390000
%%%%%%%%%%%%                                                        00391000
        WHERE STATUS-PROP EQL 0                                     00392000
        KEY                                                         00393000
              ( PAPEL-PROP                                          00394000
               ,CODINS-PROP                                         00395000
              )                                                     00396000
       ,INDEX SEQUENTIAL                                            00397000
       ,DUPLICATES                                                  00398000
      ;                                                             00399000
                                                                    00400000
SETPROPPUCOTASC   SUBSET OF DSPROP                                  00401000
%%%%%%%%%%%%%%%                                                     00402000
        WHERE STATUS-PROP EQL 0                                     00403000
        KEY   ( PAPEL-PROP                                          00404000
               ,PU-PROP     ASCENDING                               00405000
               ,COT-PROP    ASCENDING                               00406000
               ,CODINS-PROP                                         00407000
              )                                                     00408000
       ,INDEX SEQUENTIAL                                            00409000
       ,DUPLICATES                                                  00410000
      ;                                                             00411000
                                                                    00412000
SETPROPPUCOTDES   SUBSET OF DSPROP                                  00413000
%%%%%%%%%%%%%%%%                                                    00414000
        WHERE STATUS-PROP EQL 0                                     00415000
        KEY                                                         00416000
              ( PAPEL-PROP                                          00417000
               ,PU-PROP     DESCENDING                              00418000
               ,COT-PROP    DESCENDING                              00419000
               ,CODINS-PROP                                         00420000
              )                                                     00421000
       ,INDEX SEQUENTIAL                                            00422000
       ,DUPLICATES                                                  00423000
      ;                                                             00424000
```

ANEXO F - DESCRIÇÃO DO BANCO DE DADOS DBOFPUB NO FORMATO DASDL

I went to my first programming school in 1963.
It was for IBM Autocoder.
I was told at that class that COBOL was dead!
Hannah Moyer Salvatore

A estrutura básica de um programa COBOL é composta por quatro divisões. A IDENTIFICATION, contendo características básicas sobre o programa; a ENVIRONMENT, que fornece informações relativas ao ambiente de processamento; a DATA que contém a descrição dos tipos de dados, entradas e saídas manipulados pelo programa e finalmente a PROCEDURE, onde são encontradas todas as instruções para leitura, processamento e saída de informação [UNIS96a, UNIS96b, UNIS96c, UNIS97a].

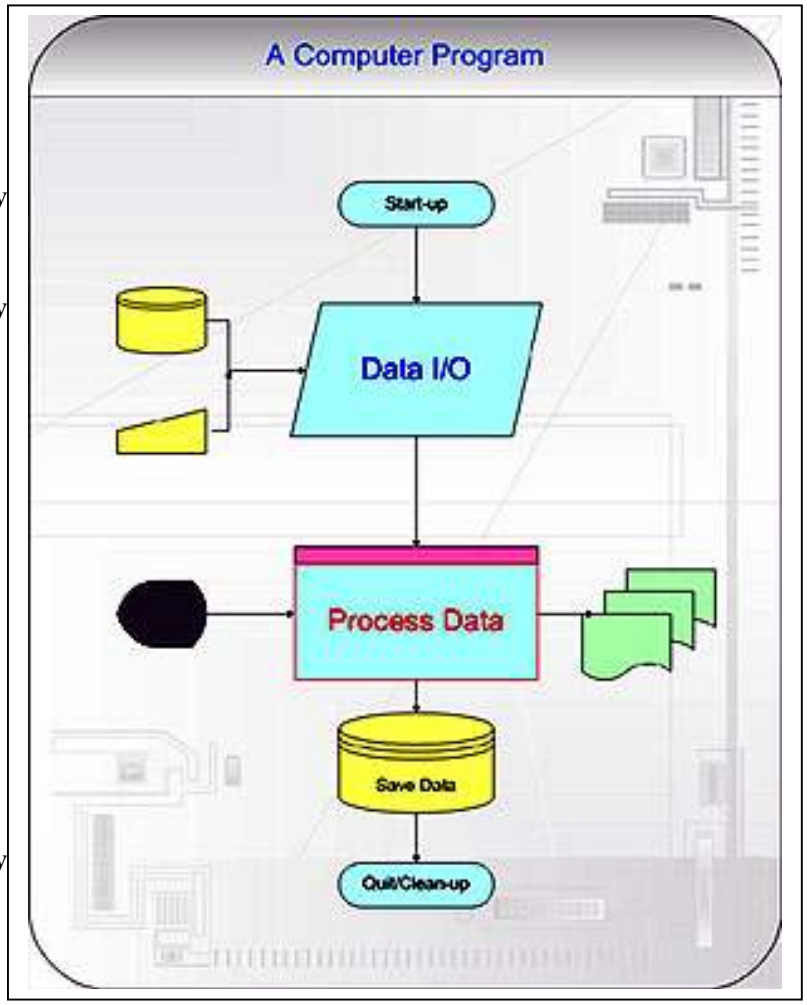

Figura F.1 **– Fluxo de um programa e seu paralelo com as divisões do COBOL (Fonte: [KIM02]).**

A IDENTIFICATION DIVISION identifica o programa fonte, permitindo ao usuário inserir informações como o nome do programa e a data em que ele foi escrito, conforme consta no código abaixo:

```
000010 Identification Division.
000020 Program-ID.&&& MyProgram.
000030 Author. Jane Doe.
000040*-----------------------------------------
000050* This is a killer program, It does everything
000060*-----------------------------------------
000070 Date-Written.& 05/01/2002.
000080 Date-Compiled. 05/02/2002.
000090 Security. None.
000100 Installation.& Test system.
```

As colunas de 1 a 6 são ocupadas pela numeração de linhas que, apesar de não ser obrigatória, é altamente recomendável. Os comentários podem ser ineridos a partir de um asterisco na coluna 7. Excetuando os comentários, todas as declarações em COBOL precisam terminar com um ponto. Em alguns casos alguns comandos podem ser agrupados em um mesmo parágrafo. Neste caso o ponto é necessário apenas ao final do parágrafo.

A ENVIRONMENT DIVISION especifica um método padrão para expressar os aspectos de dependência que o programa pode apresentar em relação às características físicas de um equipamento, tais como configuração da compilação e do código objeto gerado, além de informações relativas aos controles de entrada e saída. Nesta divisão, o compilador COBOL é informado sobre quais recursos serão requisitados pelo programa (arquivos de dados, periféricos especiais, modos de comunicação entre os programas) A INPUT-OUTPUT SECTION lista os arquivos de entrada e saída requisitados pelo programa. No exemplo abaixo, *MyFile* é um arquivo de acesso indexado, identificado pela chave *PrimKey*.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-PC.
  OBJECT-COMPUTER. IBM-PC.
SPECIAL-NAMES.
  CRT STATUS IS KEY-STATUS
  COMMAND-LINE IS LDA
  CURSOR IS CUR-POS
  CONSOLE IS WS
```

```
  call-convention 74 is WINAPI.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    Select MyFile& assign to disk
       organization indexed record key PrimKey
       access mode dynamic
       lockmode is automatic
       file status is f-st.
```

A DATA DIVISION aloca espaços de memória para os itens de dado do programa. A FILE SECTION detalha cada arquivo listado em FILE-CONTROL da ENVIRONMENT DIVISION. A WORKING-STORAGE SECTION traz a definição das variáveis, constantes e parâmetros utilizados no programa.

```
DATA DIVISION.
FILE SECTION.
FD MyFile  Label records standard,
           value of file-id is FMyFile.
01  MyRec.
    02 PrimKey    pic x(45).
    02 MiscStuff  pic x(256).

FD YourFile& Label record standard.
01 YourRec        pic x(126).

WORKING-STORAGE SECTION.
77  FMyFile       pic x(50).     *> File name SET dynamically

01  F-ST.                        *> File status
  02 F-ST1        PIC X.
  02 F-ST2        PIC X.
  02 F-STA        REDEFINES F-ST2 PIC 9(2) COMP.
01  F-STB         PIC 9(4).

01  CUR-POS.                      *> Cursor position - column and
row
  03 CUR-ROW      PIC 999 VALUE 1.
  03 CUR-COL      PIC 999 VALUE 1.

01  KEY-STATUS.                   *> User key types - mouse controls

  03 KEY-TYPE     PIC X.
  03 KEY-CODE-1   PIC 99 COMP-X.
  03 KEY-CODE-1Z  REDEFINES KEY-CODE-1 PIC X.
  03 KEY-CODE-2   PIC 99 COMP-X.
```

Na PROCEDURE DIVISION os comandos são listados e executados, arquivos são abertos e fechados, registros são lidos e manipulados, informações são apresentadas para o usuário, impressas e armazenadas em algum dispositivo. Cada divisão é composta por seções, que por sua vez são compostas por parágrafos, que são compostos por sentenças, que finalmente são compostos por comandos. Estes conceitos são um reflexo da tentativa de simular uma linguagem natural atendendo a um dos requisitos para a concepção do COBOL. [GAVI00]

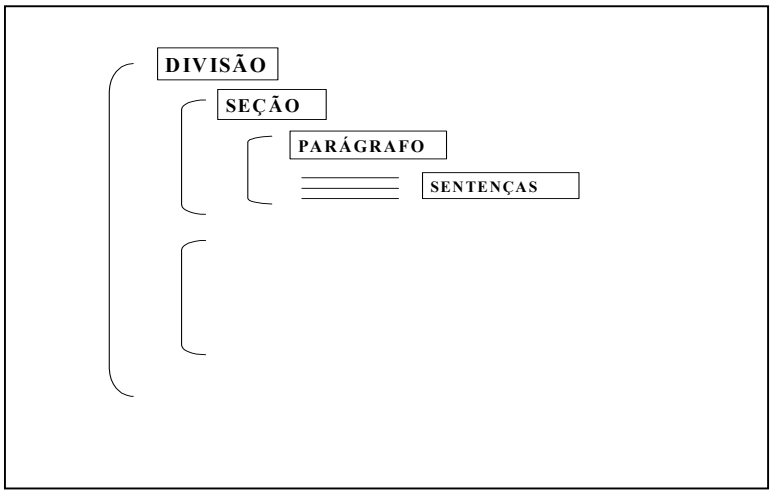

Figura F.2 **– Estrutura hierárquica de um programa COBOL.**

Críticos da linguagem, alegam que escrever programas em COBOL é uma tarefa árdua e trabalhosa devido ao número de comandos necessários para se realizar procedimentos simples. Esta afirmação é questionável na medida em que o vasto vocabulário de COBOL permite ampla documentação do fonte em língua inglesa e muitas das declarações são opcionais, existindo apenas para fornecer maior clareza de leitura ao fonte.