RICHARD FAUST

SOFTWARE COMO INTERPRETAÇÃO: UMA ESTRATÉGIA DE SOFTWARE CENTRADA NO REGISTRO LINGÜÍSTICO DOS USUÁRIOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre.
Curso de Pós-Graduação em Engenharia de Produção,
Universidade Federal de Santa Catarina.
Orientador: Prof. Leila Amaral Gontijo

FLORIANÓPOLIS

1995

RICHARD FAUST

SOFTWARE COMO INTERPRETAÇÃO: UMA ESTRATÉGIA DE SOFTWARE CENTRADA NO REGISTRO LINGÜÍSTICO DOS USUÁRIOS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina.

Prof. Osmar Possamai, Dr.
Coordenador do Curso

Banca Examinadora

Prof. Leila Amaral Gontijo, Dr.
Orientador

Prof. Leonor Scliar-Cabral, Dr.
Examinador Externo

Prof. Walter de Abreu Cybis, Dr.

Resumo

Abordagens ergonômicas ao desenvolvimento de software vêm tentando incorporar savoir-faire ergonômico nos métodos da ciência da computação, a fim de projetar uma melhor interação homem-computador. Estas tentativas são geralmente baseadas em conhecimento de psicologia cognitiva ou relacionado a análise da tarefa. Estudos recentes do fenômeno software de uma perspectiva semiótica lançaram uma nova luz neste assunto. O aspecto comunicacional do software, seja em construção ou durante o uso, pode se beneficiar de uma perspectiva semiótica que auxilia na explicação e predição do funcionamento da comunicação. Este trabalho descreve uma abordagem de embasamento semiótico. Ela consiste da elicitação e organização de conhecimento sobre a linguagem profissional do usuário para uso no levantamento dos requisitos do software e no projeto da interface. Desta maneira os dois aspectos relacionados à interação homem-computador são tratados: utilidade (requisitos do sofware) e usabilidade (interação homem-computador). Uma estratégia de desenvolvimento de software é proposta, acoplada a um já existente método de desenvolvimento de software orientado a objetos. Dois estudos de caso ilustrativos (uma banca de revistas e uma escola de idiomas) são descritos, fornecendo resultados preliminares e sugerindo novas pesquisas.

Abstract

Ergonomic approaches to software development have been trying to incorporate ergonomic savoir-faire into Computer Science methods, in order to design a better human-computer interaction. These attempts are usually based on Cognitive Psychology or Task Analysis related knowledge. Recent studies of software phenomena from a Semiotic perspective shed a new light on this subject. The communication aspect of software, either under construction or during use, can benefit from a Semiotic perspective that helps to explain and predict the working of communication. This article describes one such Semiotic based approach. It consists on the gathering and organization of user professional language knowledge for use in software requirement elicitation and user interface design. In this way both aspects of human computer interaction are dealt with: utility related (software requirements) and usability related (user-computer interaction) aspects. A software development strategy is proposed, linking to an object-oriented software development method. Two illustrative case studies (a news-stand and a language school) are described, furnishing preliminary results and directions for further research.

Sumário

1. Introdução

A Ciência da Computação vem experimentando uma mudança de paradigma[1], onde o objeto de estudo não é mais apenas o componente computacional (hardware ou software). As pessoas que interagem com estes componentes passaram a ser também parte do objeto de estudo. Na Engenharia de Software isto acontece quando, ao invés de adotarmos uma abordagem dirigida ao produto (software), nos preocupamos com o processo (construção de software) (Floyd, 1988).

A construção do software pode ser vista como composta de duas dimensões básicas: a das pessoas que fazem o software (dimensão interna) e das pessoas que usam o software (dimensão externa) (Meyer, 1988); sendo portanto uma atividade basicamente humana.

Para investigar estas dimensões humanas do software as ferramentas comumente usadas tais como formalismos (vindos da Matemática) ou métricas e métodos (inspirados nas engenharias), não são mais suficientes. Precisamos de idéias e instrumentos de ciências que já estudem atividades do ponto de vista das pessoas que participam do processo, e não apenas do produto final.

Uma visão possível é a da Semiótica, a partir da qual pode-se tratar dos aspectos de comunicação envolvidos na construção e uso de programas. Esta é a perspectiva adotada neste trabalho, onde o software é visto não como um objeto dotado de propriedades mas sim como um fenômeno no qual pessoas interpretam manifestações de um programa de computador como software em dados contextos.

A partir desta perspectiva pode-se tratar o desenvolvimento de software como construção de significado e redefinir os papéis do analista e usuário. Estes passam a ser tratados como produtores e consumidores de material simbólico, respectivamente. Este material simbólico é enviado através da mídia computador e interage com outros sistemas de significação já existentes, como, por exemplo, o registro lingüístico relacionado ao trabalho do usuário. Desta perspectiva originam-se os objetivos e a metodologia descritos abaixo.

1.1. Objetivos

Objetivo Geral: Explorar uma estratégia de desenvolvimento de software centrada no registro lingüístico do usuário através de estudos de caso.

---

[1] A acepção de "paradigma" usada aqui é aquela da Filosofia da Ciência (Kuhn, 1970).

Objetivos Específicos:

- Estabelecer relações entre registro(s) e software, na construção e uso de programas de computador.

- Incrementar e desenvolver técnicas para levantamento do conhecimento sobre o registro lingüístico.

- Investigar formas de (re)uso do conhecimento sobre o registro na construção de programas.

1.2. Metodologia e Organização desta Dissertação

O presente trabalho pode ser dividido em duas partes: na primeira o estabelecimento da relação entre Semiótica e Software é abordado e são estabelecidas as hipóteses que orientaram a segunda parte, onde são definidos instrumentos e descritos aspectos relevantes dos estudos de caso. Esta divisão reflete a metodologia usada no trabalho.

O capítulo 2 trata de citar algumas abordagens de Ergonomia de Software para situar a presente em relação a elas. A disciplina da Semiótica Computacional é brevemente descrita no capítulo 3, onde a abordagem deste trabalho é situada no contexto das pesquisas efetuadas nesta área. A definição de software como construção de significado é o tópico do capítulo 4, onde também são exploradas a relação de software com registro. Este capítulo finaliza a primeira parte.

A segunda parte começa com o capítulo 5, onde o Léxico Ampliado da Linguagem, uma estrutura associada ao levantamento de informações sobre o registro é descrita. O DOM, um modelo do registro relacionado ao reuso das informações sobre o registro na construção de programas é discutido no capítulo 6. Dois estudos de caso realizados são relatados no capítulo 7. A discussão dos resultados, de trabalhos relacionados e da continuação das pesquisas é o assunto do capítulo 8. Ao final de cada capítulo existe uma seção denominada Discussão, onde os tópicos apresentados são discutidos no contexto do presente trabalho.

2. Ergonomia e Software

A Ergonomia de Software é uma disciplina relativamente jovem e que, portanto, ainda busca bases estáveis sobre as quais desenvolver seus trabalhos. Sendo a Ergonomia, sobretudo, a aplicação válida de conhecimentos sobre o homem e seu trabalho para melhorar as condições de trabalho (Scapin, 1993), é preciso determinar as fontes deste conhecimento. Tendo sido estas determinadas, são estabelecidos métodos para sua aplicação de forma válida.

Este capítulo procura, primeiramente, evidenciar algumas abordagens existentes e suas respectivas bases de trabalho. Em seqüência procura-se localizar a abordagem utilizada neste trabalho em relação aos estudos relatados.

2.1. Abordagens de Ergonomia de Software

Para que um determinado fenômeno possa ser estudado por um campo científico qualquer, é preciso que este fenômeno seja caracterizado como pertinente ao objeto de estudo daquele campo científico. É só então que o conhecimento existente e os métodos de descrição ou predição de fenômenos podem ser aplicados.

A Ergonomia de Software tem procurado estabelecer as bases para o estudo da interação homem-computador através da caracterização desta como objeto de estudo de diversas áreas, tais como a Psicologia Cognitiva e Semiótica/Lingüística. Esta caracterização possibilita então que os conhecimentos e métodos destas áreas possam ser utilizados na avaliação e concepção de software interativo. Abaixo, alguns exemplos:

Barthet (1988) tem sua abordagem baseada em uma crítica aos métodos tradicionais de concepção de software, geralmente executados por analistas de sistemas, onde o ponto de vista do usuário não é levado em conta. Nestes métodos tradicionais, segundo ela, a concepção do diálogo homem-computador é deduzida da lógica do sistema, de seu funcionamento e não da lógica do usuário, de utilização no posto de trabalho. Para ela, as melhorias no hardware e no software básico permitem o surgimento de sistemas melhor adaptados aos usuários, sendo o maior entrave os métodos de concepção. Como forma de abordar este problema é proposto o uso da Psicologia Cognitiva e Ergonomia para melhorar a comunicação homem-máquina, atuando junto às representações externa (que é o software tal como será visto e manipulado pelo usuário) e conceitual (ligada ao analista). A partir disto são propostos métodos de concepção e de avaliação.

Coutaz (1990) também se preocupa com a comunicação homem-computador e com abordagens da Psicologia Cognitiva. Ela constata que os conhecimentos existentes "não permitem, atualmente, a determinação científica das características cognitivas do usuário" (Coutaz, 1990, p. 3). Seu trabalho procura organizar e estruturar os métodos quantitativos (formais, porém simplistas), qualitativos (muito informais) e recomendações

práticas (às vezes contraditórias). No entanto, seu modelo PAC procura conciliar as vantagens do modelo linguagem com o modelo multiagente, dois modelos mais relacionados com a arquitetura do software do que com os aspectos cognitivos da interação.

Em resumo, as abordagens usadas como exemplo, tendem a enfocar o problema da comunicação a partir de um ponto de vista da Psicologia Cognitiva. Isto será retomado na discussão no final do capítulo.

Um exemplo de abordagem que utiliza uma metáfora lingüística pode ser encontrado em (Foley & Van Dam, 1982). O Modelo de Linguagem proposto por estes autores divide os componentes da interface em duas linguagens: "com uma, o usuário se comunica com o computador; com a outra, o computador se comunica com o usuário" (Foley & Van Dam, 1982, p. 220). Este tipo de abordagem assume, implicitamente, uma analogia da interação homem-computador com o diálogo entre duas pessoas. Cada uma destas linguagens é decomposta em quatro partes: conceitual (objetos do domínio do problema), semântica (funcionalidade em detalhes), sintaxe (seqüência de entradas e saídas) e léxica (como as entradas e saídas são formadas a partir de primitivas).

Este é um exemplo de aplicação da lingüística ao projeto de interfaces. Na discussão retomaremos este exemplo, mostrando como a abordagem deste trabalho utiliza uma fundamentação lingüística, porém de forma diferente.

No que diz respeito à concepção/avaliação de interfaces, pode-se dividir os aspectos, grosso modo, em relacionados ao "diálogo" e à compatibilidade com a tarefa.

Os aspectos relacionados ao diálogo têm sido tratados com recomendações do tipo "faça isso" ou "não faça isso". Mais recentemente alguns autores tais como (Scapin, 1993) e (de Souza, 1993), tem procurado sintetizar ou gerar este mesmo tipo de conhecimento em princípios e critérios genéricos.

Quanto à compatibilidade com a tarefa, Barthet (1988) propõe uma abordagem que leva em conta desde o início da concepção a análise da tarefa, a fim de promover a compatibilidade com esta. Já a proposta de Valentin & outros (1993), define pontos dentro do ciclo de desenvolvimento de um software onde são feitas avaliações ergonômicas com protótipos e amostras da população de usuários. Estas avaliações servem, então, para redirecionar os desenvolvimentos subseqüentes.

Na discussão a seguir procuraremos sinalizar como estas questões são abordadas dentro da estratégia utilizada neste trabalho.

2.2. Discussão

No tratamento das questões ligadas à comunicação homem-computador, este trabalho difere dos trabalhos relatados, em alguns pontos:

- a abordagem tem suas bases na Semiótica/Lingüística e não na Psicologia Cognitiva, como os trabalhos relatados em Barthet (1988) e Coutaz (1990).

- a analogia do diálogo entre duas pessoas para o que acontece com uma pessoa e um computador é em parte abandonada. Utiliza-se, ao invés, a perspectiva do computador como uma mídia interativa, mais do que como parceiro em um diálogo.

- o que se propõe é uma abordagem baseada no registro. **Registro é a linguagem utilizada pelo usuário.** Utilizando-se uma linguagem apropriada, pretende-se melhorar a qualidade da comunicação. Esta definição será refinada no decorrer do trabalho.

Quanto aos aspectos de diálogo e compatibilidade com a tarefa:

- os aspectos do diálogo procuram ser tratados, parcialmente, pela adoção de uma linguagem adequada ao registro do usuário.

- a compatibilidade com a tarefa é buscada através do conhecimento de como o usuário organiza seu trabalho (por meio da linguagem) e de um trabalho de prototipação para os outros ajustes e exige uma preocupação psicolingüística e sociolingüística. Apesar de necessitar de estudos mais aprofundados, esta combinação pode ser uma alternativa à Análise da Tarefa com entrevistas e observações.

Além disso, não é somente a comunicação quando do uso de um software que é importante. A comunicação que se dá entre analista e usuário é fundamental para que o analista defina os requisitos apropriados que o software deve satisfazer. Aqui entra novamente a abordagem centrada no registro do usuário.

A melhoria da comunicação no fazer o software e, conseqüentemente, da sua funcionalidade, faz com que a interface do software melhore. Conceituações mais recentes do que é 'interface' mostram que a utilidade é tão importante quanto a usabilidade: não basta lidar com as metas de interação do usuário, mas também com suas metas fundamentais (Scapin, 1993). Não é suficiente nos preocuparmos com o acesso a um conjunto de funcionalidade se este é o conjunto errado, não correspondendo às expectativas do usuário em relação ao software. Este ponto específico será retomado no capítulo 4. Software como Construção de Significado, e a discussão geral será retomada no capítulos 3. Semiótica e Computadores e 4. Software como Construção de Significado. Os demais capítulos se dedicarão a estabelecer instrumentos e a relatar e discutir resultados preliminares de uma estratégia centrada no registro.

3. Semiótica e Computadores

"Semiótica é a ciência que estuda a vida dos signos no seio da sociedade."

Curso de Lingüística Geral por Ferdinand de Saussure, pg. 24

"Um signo é tudo que pode ser tomado como substituindo significativamente outra coisa."

A Theory of Semiotics por Umberto Eco, pg 7

A perspectiva do computador como mídia, através da qual pessoas se comunicam, vem sendo cada vez mais aceita e usada como base para o entendimento dos sistemas computacionais. A idéia de que o computador é usado através de signos, e de que o projetista propõe signos quando cria um software, gera um novo campo de pesquisa. Este campo de pesquisa é o da Semiótica Computacional, que vem se desenvolvendo nos últimos anos em várias frentes e com diferentes abordagens.

O objetivo deste capítulo é dar um panorama de como a Semiótica pode ser aplicada aos sistemas computacionais; e definir conceitos que possibilitarão o desenvolvimento do capítulo 4. Software como Construção de Significado.

3.1. Semiótica Computacional

A Semiótica vem sendo aplicada a vários tipos de mídia, e essa aplicação tem resultado em um melhor entendimento da natureza destas mídias e de como se pode trabalhá-las de forma efetiva. O cinema e o teatro são exemplos de mídias analisadas sob uma perspectiva semiótica.

As mídias cinema e teatro são primariamente meios de expressão artística, onde um uso "efetivo" e direto pode não ser imprescindível, enquanto que o computador é usado em situações de trabalho, onde compreensão de forma inapropriada pelo usuário compromete o uso do software: "Apesar de ter fortes similaridades com arte, IHC no entanto diverge desta consideravelmente já que tipicamente não se espera que as mensagens computacionais elicitem muitas emoções nos usuários, mas sim um reduzido número de alternativas racionais para cada mensagem." (de Souza, 1993, p. 756)[2]

---

[2]Obviamente, isto é verdadeiro se o computador estiver sendo usado em uma situação de trabalho, como meio para se desempenhar algum tipo de trabalho. E deixa de ser verdadeiro se o computador for usado como mídia para alguma manifestação artística.

Cada aspecto de um computador, desde a linguagem de mais baixo nível até a interface com o usuário, funciona como signo para alguém (Andersen, 1992): a interface de um sistema de reserva de passagens de avião representa assentos e reservas para o atendente; o texto de um programa representa possíveis execuções de um programa para o programador; o compilador é um meta-signo que, à semelhança de uma gramática, define que textos de programa são válidos; e assim por diante.

Esta dimensão semiótica dos sistemas computacionais é o que a Semiótica Computacional se propõe a abarcar, enquanto ramo da Semiótica que "estuda a natureza específica dos signos baseados em computador e como eles funcionam em uso." (Andersen, 1990, p. 3)

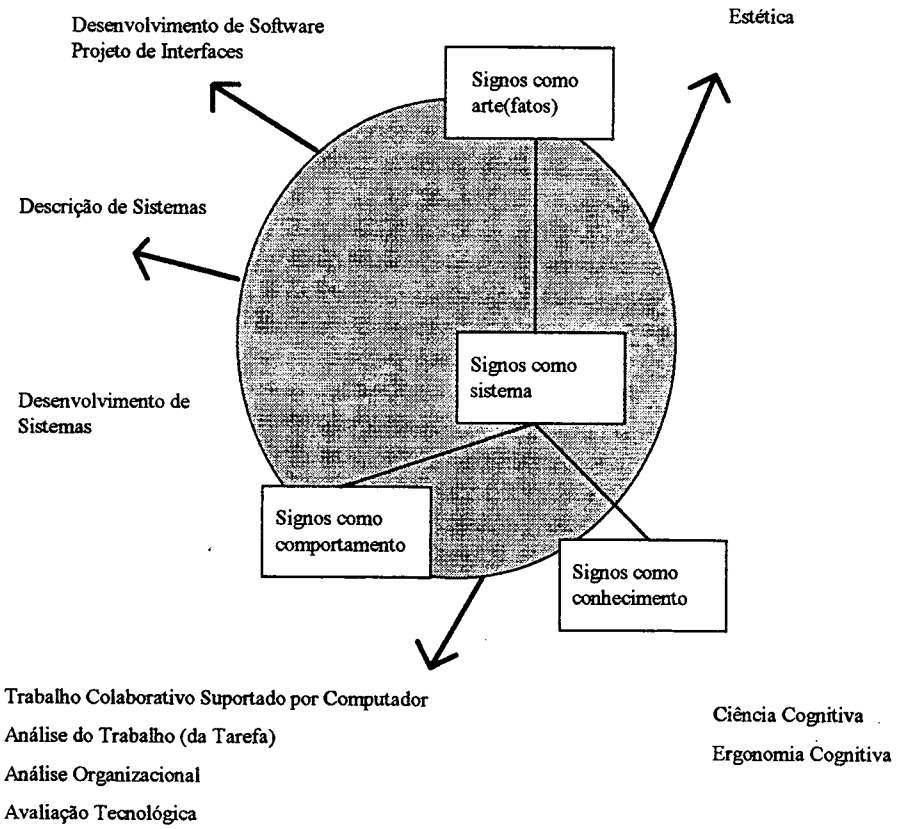

Figura 2.1. Mapa da Semiótica Computacional (Andersen, 1990, p. 18)

O mapa acima é uma tentativa de estabelecer o escopo da Semiótica Computacional, mostrando o que ela abrange, quais são suas fronteiras e as possibilidades de colaboração com outras áreas. É importante notar que esta perspectiva semiótica é apenas uma das possíveis, e que diversos aspectos de um sistema computacional ficam fora de seu escopo, necessitando da concorrência de outras áreas para uma abordagem holística.

O aspecto central é o de signos como sistema, onde "o indivíduo é considerado como um criador, intérprete e referente de signos, como um usuário ou reprodutor de

potencial de significado e código comuns, utilizando os resultados de trabalho semiótico feito por outros." (Andersen, 1990, p. 19)

Dos outros aspectos, o de signos como comportamento é um que ocupa um papel bastante importante num estudo semiótico dos computadores, já que está relacionado com como signos são usados para coordenar ações. Winograd & Flores (1985) nos dão um exemplo de abordagem centrada neste aspecto. Nele os autores usam a teoria de atos da fala para propor um gênero de sistema computacional que suporte os padrões recorrentes de comunicação dentro das organizações (e que constituem, ou suportam, a existência e funcionamento destas).

Signos como conhecimento é um aspecto que geralmente traz a necessidade do tratamento por outras áreas, que tratem dos aspectos cognitivos. Os trabalhos de Barthet (1988) e Coutaz (1990), citados no capítulo 2, são exemplos deste enfoque, onde o que é importante são os processos cognitivos envolvidos com o software, e não o aspecto de produção de signos. de Souza (1993) sugere que estas abordagens podem ser complementares, com a abordagem cognitiva engendrando o conteúdo das mensagens e a semiótica a expressão destas. Ou, ainda, que a Engenharia Cognitiva estabelece metas de compreensão e uso produtivo de sistemas e que a Engenharia Semiótica procura definir como produzir sistemas como mensagens enviadas do projetista ao usuário para alcançar estas metas.

O último aspecto é o de signos como arte(fatos). Nele "o indivíduo é considerado como um inovador de potencial de significação e código, como um explorador e inventor de signos" (Andersen, 1990, p. 19). Esta exploração dos limites dos códigos em uma mídia leva ao estabelecimento de novas formas de produção sígnica, que podem ser então reaproveitadas mesmo em atividades cujo aspecto artístico não seja o essencial, tal como no trabalho. A arte feita no computador não é apenas intrinsecamente importante, mas também traz contribuições para o uso de computadores em geral.

Na próxima seção o aspecto essencial de signos como sistema será retomado com a definição de conceitos que permitem organizar e entender estes sistemas.

3.2. Elementos de uma Semiótica Computacional

A fim de trabalharmos com signos como sistema, é preciso estabelecermos o que é sistema, para então definirmos os conceitos que servem para estudá-los.

O próximo item consiste de uma discussão para estabelecer que acepção de sistema (ou esquema) está sendo usada na Teoria de Semiótica Computacional de (Andersen, 1990). Na seqüência, conceitos para lidar com estes sistemas são sumarizados.

3.2.1. Esquema Semiótico

Sistemas podem ser considerados como artefatos que podemos construir, ou como estruturas onde registramos regularidades de uma práxis (Andersen, 1990). Na Semiótica, o primeiro caso, de artefatos passíveis de construção, dificilmente pode ser verdade, pois seria algo como projetar o conteúdo e expressão de uma língua e fazer com que esta fosse falada pelas pessoas e usada nas suas atividades.

Nos resta o segundo caso, onde o sistema existente é registrado e desenvolvem-se "novas práticas através de experimentação, registro das regularidades subjacentes às novas práticas, e projeto do sistema para suportar aquelas regularidades que são desejáveis" (Andersen, 1990, p. 121).

Se o sistema em questão for de signos baseados em computador, o projetista estará em um papel mais próximo de um escritor, autor teatral ou mesmo de um arquiteto, que propõe signos, do que o de um engenheiro que constrói artefatos. Esta discussão será retomada e expandida no capítulo 4, Software como Construção de Significado.

A Teoria de Semiótica Computacional de Andersen trabalha com uma noção de esquema semiótico (o termo sistema fica reservado aos aspectos paradigmáticos do esquema) que tem uma relação dialética com o uso dos signos: "Esquema e uso são interdependentes: não haverá uso de signos sem um esquema comum, já que o uso individual de signos ganha seu significado do esquema semiótico social, respeitando-o ou desviando dele. Por outro lado, não haverá esquema sem uso, já que o esquema são as características invariantes do uso." (Andersen, 1990, p. 123) A definição a seguir resume este ponto de vista:

> Um *esquema semiótico* é uma descrição dos *invariantes* do uso de signos que são *socialmente* possíveis em um dado *tipo de situação* por uma dada *comunidade* de usuários de signos. Um esquema semiótico descreve uma forma a qual é uma *semiótica*, uma semiótica sendo um sistema e um processo, cada um dos quais pode ser dividido em dois planos, de expressão e de conteúdo, cujos elementos comutam. (Andersen, 1990, p. 123)

Embora esta definição seja similar às feitas por outros autores, um aspecto específico é sua associação a tipos de situação. Isto é particularmente importante para os esquemas semióticos computacionais, que geralmente estarão relacionados ao uso do computador em algum tipo de situação.

As implicações práticas desta noção de esquema são discutidas na seção 3.4. O próximo item define conceitos relacionados a esta definição de esquema semiótico.

3.2.2. Conceitos Relacionados

Abaixo temos um sumário dos principais conceitos utilizados na definição de esquema semiótico:

Invariantes

Invariantes estão relacionados ao significado. Quando um elemento é substituído por outro e esta troca resulta em troca de significado podemos dizer que encontramos um invariante. Este distingue-se da realização, que pode conter elementos variáveis, como por exemplo, em uma dada língua, aqueles associados a um sotaque, mas que não levam à troca de significado. Eles nos permitem enfocar as características realmente importantes de um sistema sem nos perdermos nos detalhes.

Tipo de Situação[3]

> "Uma *situação* é um conjunto de atos comunicativos e não-comunicativos que têm lugar em um tempo e lugar específicos. ... deve ser concebida como uma unidade por alguma comunidade de falantes, ter um nome especial, ter as fronteiras delimitadas, etc." (Andersen, 1990, p. 54)

> "Uma situação pode pertencer a um ou mais *tipos de situação*. Um tipo de situação é caracterizado pelos papéis de seus participantes, suas tarefas e metas, ..." (Andersen, 1990, p. 123)

A noção de tipo de situação será retomada na discussão deste capítulo, a fim de refinar a noção de registro, introduzida no capítulo anterior.

Forma/Substância e Expressão/Conteúdo[4]

Estas duas dicotomias são básicas para a definição de signo na visão estruturalista:

---

[3]Este conceito pode ser encontrado na obra do etnógrafo da fala Dell Hymes (Hymes, 1968)

[4]A elaboração destas dicotomias se deve, primeiramente, ao lingüista L. Hjelmslev (Hjelmslev, 1975)

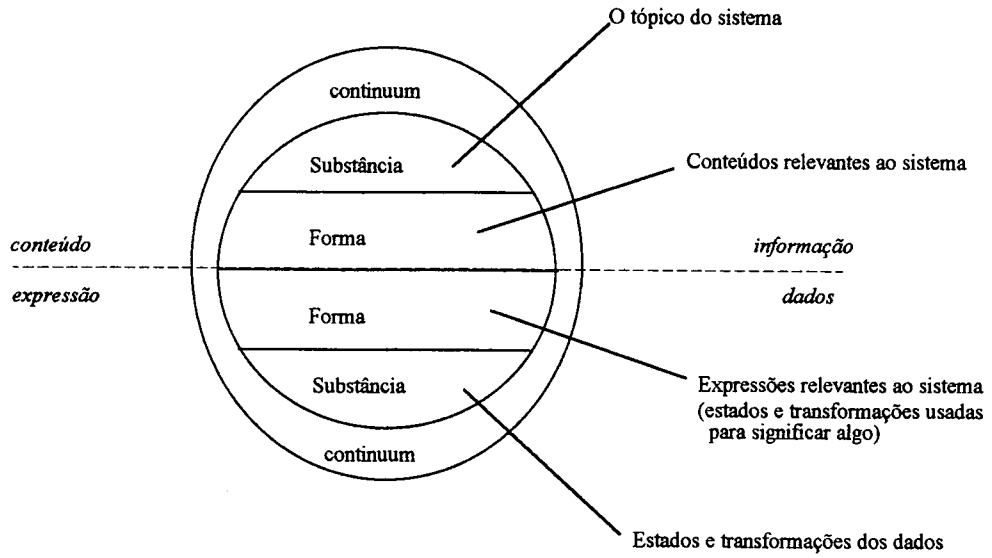

Figura 2.2. O conceito estruturalista de signo (Andersen, 1992, p. 18)

O exemplo abaixo, baseado em Andersen (1990, pp. 69-70), procura clarificar a visão esquemática da figura 2.2. :

> Quando uma moeda é jogada, isto pode ser um ato simbólico ou não. Caso eu esteja apenas querendo me livrar dela, não há nada mais do que isso: eu apenas jogo a moeda. No entanto, eu posso usar este ato para resolver alguma discussão. Para isso serão escolhidas que posições da moeda são relevantes (forma da expressão) de um *continuum* de todas as posições possíveis da moeda (substância da expressão). Assim, a expressão é o plano observável e identificável, que entra em contato com os sentidos, o que é concreto (Dubois e outros, 1991). Às posições relevantes da moeda serão associados elementos do plano da conteúdo. No caso de a disputa em questão ser a de um adulto e uma criança decidindo quem vai lavar os pratos, a substância do conteúdo consiste de todas as maneiras de um adulto e uma criança lavarem os pratos. Como as maneiras relevantes que foram acordadas entre eles é que cada um lavaria os pratos sozinho, ficamos com dois elementos: *adulto lava* e *criança lava* que constituem nossa forma do conteúdo. Uma correlação entre os dois planos será também acordada por ambos (estabelecer um esquema semiótico para possibilitar o uso). Digamos que o que foi acordado foi o seguinte: cara-adulto lava, coroa-criança lava. Temos agora o esquema semiótico definido (Figura 2.3.).

	forma	substância
conteúdo	*adulto lava* *criança lava*	*todas as maneiras de um adulto e uma criança lavarem pratos.*
expressão	*cara* *coroa*	*todas as posições de uma moeda.*

Figura 2.3. Esquema semiótico para o exemplo da moeda e da lavação de pratos

Assim, signos não existem como entidades físicas, mas como correlações entre elementos de dois planos distintos, da expressão e do conteúdo. É mais apropriado, portanto, falarmos de funções-signo e não de signos (Eco, 1979). No entanto, o termo signo é bastante útil e já consagrado, e é mantido para fins retóricos por vários autores, inclusive Umberto Eco.

Os conceitos de forma/substância e expressão/conteúdo são básicos à Semiótica de vertente estruturalista e são subjacentes à abordagem adotada neste trabalho, especialmente na relação entre tipo de situação de uso de signos e a caracterização da forma do conteúdo e da expressão.

Teste da Comutação

Para identificar que elementos são pertinentes e/ou deverão pertencer (no caso de estarmos prevendo alterações ou propondo um novo) a um esquema semiótico, a ferramenta utilizada é a do Teste da Comutação. A substituição de um elemento em um dos planos (expressão ou conteúdo) pode ou não acarretar alteração no outro plano. Caso isso ocorra, o teste é bem sucedido e foi encontrado um elemento pertinente. No exemplo das moedas e da lavação de pratos, caso troquemos cara por coroa (plano da expressão) teremos uma mudança correspondente no plano do conteúdo: de *adulto lava* para *criança lava*.

Este teste é uma maneira de levantarmos informação sobre as relações entre os termos e não quanto aos termos em si. Os elementos que passam no teste são chamados de *invariantes*. Estes diferem das realizações (variantes) que são aspectos que podem variar, mas que não trazem alteração de significado: por exemplo, usar uma moeda de 25 ou de 50 centavos.

Sistema e Processo

Quando efetuamos o Teste da Comutação, precisamos trocar unidades presentes por outras que não estão presentes. Estas relações *in absentia*, também

chamadas de paradigmáticas, são referentes ao **sistema**. Já as relações entre elementos que co-ocorrem, *in praesentia*, são chamadas de sintagmáticas e dizem respeito ao **processo** (Scliar-Cabral, 1993).

Os elementos do sistema contraem uma relação *ou* entre eles. Um exemplo seria quando alguém diz "mola" e não "bola" porque o significado seria alterado (Scliar-Cabral, 1993). Embora o /b/ não esteja presente, é através das suas relações de negação com outros elementos que o /m/ adquire seu valor (o /m/ não é o /b/, não é o /p/, e assim por diante) . Diz-se que o /b/ e o /m/ pertencem ao mesmo paradigma, já que podem ocorrer no mesmo contexto. Vale ressaltar que isto vale não só para os fonemas, mas para outras unidades funcionais em relação ao significado.

Já os elementos do processo contraem uma relação *e*. Esta relação se dá em Lingüística, por exemplo, pela aplicação desta conjunção lógica para a descrição da construção das palavras e dos enunciados, com elementos que não ocorrem no mesmo contexto, mas sim em seqüência e *in praesentia*. O processamento necessário para o texto "Pare", por exemplo, é p + a + r + e (Dubois e outros, 1991).

Neste item procurou-se revisar alguns conceitos elementares de Semiótica, relacionados ao esquema semiótico e desenvolvidos dentro da vertente teórica do estruturalismo. Estes conceitos permitem descrever vários aspectos de um esquema, incluindo o conteúdo que ele articula, como ele se expressa, quais os elementos pertinentes, que tipos de relação eles contraem. Na seção 3.4. Discussão, aspectos deste trabalho relacionados com estes conceitos serão abordados. A próxima seção trata de possíveis contribuições da Semiótica ao estudo de computadores e relata alguns trabalhos nesta área.

3.3. Contribuições da Semiótica

A partir da visão estruturalista brevemente apresentada na seção anterior, ou de uma outra teoria de semiótica computacional, pode-se abordar os sistemas computacionais e contribuir para sua compreensão.

Podemos dividir os trabalhos realizados nesta área em dois grandes grupos: o primeiro é composto por aqueles que procuram entender computadores (ou software ou interface) como um fenômeno semiótico; do segundo fazem parte aqueles que investigam a interação do computador com outros esquemas semióticos.

Os itens a seguir procuram caracterizar estes grupos e exemplificá-los. Esta classificação servirá para localizar o presente trabalho dentro desta tradição emergente de pesquisa em Semiótica Computacional.

3.3.1. Caracterizar como Fenômeno Semiótico

Os trabalhos deste grupo procuram estudar o computador como mídia com características únicas. Embora possam utilizar, e assim o fazem, conhecimentos prévios de teoria semiótica e mesmo de estudos semióticos de outros meios, tratam também de caracterizar o computador como mídia com possibilidades ímpares.

(Lemos, 1991) aborda a construção de programas de uma perspectiva semiótica, onde trata do "trajeto sígnico entre idéia e programa". Ele utiliza esta perspectiva para definir três agenciamentos: formal externo (expressões formais independentes da máquina), formal de máquina (expressões com processamento em uma máquina) e usuário (propósitos e expectativas). Define várias formas de predominâncias destes agenciamentos: quando predomina o formal de máquina o usuário faz o papel de um programador e o formal externo de modelagem conceitual; quando o uso predomina, o formal externo se confunde com o formal de máquina, ambos refletindo a concepção que o usuário tem da realidade. Esta conceituação lhe permitiu reverter uma subordinação tradicional do programa (formal de máquina) em relação à especificação (formal externo), baseando a construção de programas em uma cooperação destas visões. Este deslocamento permite situar a formalização como auxílio na trajetória entre uma abstração e o programa, e não como parte principal deste processo.

A parte II. *Computers* (Andersen, 1990) procura revelar quais as semelhanças dos computadores com outros tipos de mídia e também suas características únicas. Em sua busca de uma Estilística Computacional, ele analisa os meios de expressão possíveis no computador, descrevendo os signos computacionais como compostos de três tipos de características: permanentes (geradas pelo computador, se mantêm constantes), transientes (também geradas pelo computador, mudam de acordo com o uso do signo) e manipuláveis (articulam as ações do usuário em dispositivos como mouse e teclado). Uma análise dos signos computacionais, de acordo com estas dimensões, resultou em uma tipologia, onde um dos tipos encontrados é único aos computadores: o signo interativo. Isto sinaliza na direção de que o computador como mídia tem como característica particular a interatividade, o que ajuda a estabelecer a sua estética. Além disso, ele organiza estes signos em sintagmas seqüenciais e concorrentes e procura estabelecer paralelos com outras mídias, como dança e teatro, para tratar estas questões.

(Cybis, 1994) trabalha com uma analogia ao signo computacional de (Andersen, 1990) para definir sua tipologia de OIAe (Objeto de Interação Abstrato Ergonômico). Estes objetos têm por objetivo possibilitar a portabilidade (independência de um projeto em relação a plataformas de implementação) e o raciocínio ergonômico (para a avaliação e concepção de interfaces).

Em resumo, estes trabalhos exemplificam tentativas de entender o computador como mídia através da qual se manifestam os signos computacionais. Eles objetivam estabelecer conhecimento para a concepção e análise dos esquemas semióticos baseados em computador.

3.3.2. Desvendar a Interação do Software com outros Esquemas Semióticos

Abaixo estão relacionados alguns trabalhos que abordam o projeto do software como inserido em uma realidade cultural, onde esquemas semióticos já existentes podem e devem ser considerados.

A noção peirciana de signo é adotada por Nadin (1988). Ele assume que toda interação homem-computador se dá através de signos e que a Semiótica é a opção correta para organizar e unificar o campo de projeto de interfaces. A atividade de projetar acontece dentro de uma dada cultura, onde estão imersos o projetista e o usuário, "compartilhando convenções estabelecidas e participando do estabelecimento de novos sistemas de signos, quando isto é necessário."(Nadin, 1988, p. 70).

de Souza (1993) trabalha com uma tipologia de modos de produção de signos (Eco, 1979) em uma tentativa de abordar esta mesma produção de signos, que, de acordo com esta autora, é central ao projeto de interfaces. O uso da tipologia de Eco resulta em quatro diretrizes correspondentes, que compartilham dois princípios básicos: "signos da LIU *(Linguagem da Interface com o Usuário)* devem ser sistematicamente codificados e profundamente inseridos na cultura do usuário" (de Souza, 1993, p. 766). O objetivo básico da autora é propor a Engenharia Semiótica das Linguagens de Interface com o Usuário, através do estabelecimento de um pequeno número de princípios que tenham o mesmo poder descritivo e preditivo das milhares de recomendações ergonômicas. Partindo das diretrizes baseadas em Eco, ela busca demonstrar que a qualidade relatada de diversos projetos de interface poderiam ter sido preditas pelas diretrizes. Estas estão atualmente sendo aplicadas também para propor soluções para interfaces consideradas problemáticas, por exemplo a expressão das atividades de cópia e deslocamento de arquivos no System 7.0 do Macintosh (de Souza, 1994).

Finalmente, temos o trabalho de Andersen (1990), especialmente na parte III, *Language, Work and Design*. Nesta parte o autor procura mostrar maneiras de analisar semioticamente as organizações e usar o resultado destas análises na construção de sistemas computacionais. Para tal usa, por exemplo, a idéia de campos semânticos. Um campo semântico é "um conjunto de unidades lexicais que se considera, a título de hipótese de trabalho, como dotado de organização estrutural subjacente" (Greimas&Courtés, 1979), sendo esta organização semanticamente determinada. Andersen utiliza esta noção para comparar o vocabulário (forma) com o qual os trabalhadores articulam aspectos de seu

trabalho (substância) e aquele que foi embutido nos sistemas computacionais. Constata que muitas vezes a forma refletida nos sistemas é a de indivíduos que solicitam os sistemas, mas que não serão os usuários, ou ainda baseia-se nos critérios de relevância dos projetistas, que não coincidem com aqueles dos usuários. Ele propõe como tentativa de solução o projeto de interfaces baseadas nos campos semânticos dos usuários, decorrentes de um levantamento prévio para servir de base ao projeto.

Estes trabalhos compartilham a noção da consideração da cultura e procuram estabelecer mecanismos para tratar esta interação dos esquemas semióticos baseados em computador com outros esquemas semióticos.

Obviamente, esta divisão, entre estudos do computador como mídia e da inserção cultural dos sistemas computacionais, na prática, reflete apenas uma questão de perspectiva, já que signos semióticos sempre precisarão ser expressos de alguma forma e sempre interagirão com outros esquemas semióticos. Assim, a interface de software sempre comporta estes dois aspectos, mesmo que as decisões de projeto tenham sido feitas sem levá-los em conta.

A seção seguinte comenta a influência dos conceitos definidos neste capítulo no desenvolvimento deste trabalho e fornece uma segunda definição de registro.

3.4. Discussão

Um dos conceitos deste capítulo, a noção de esquema semiótico, tem implicações práticas neste trabalho , por exemplo:

i. ter por base dados do uso real de signos por uma dada comunidade, em determinadas situações;

ii. ter em mente que esquemas semióticos, mais do que construídos, emergem do uso, e alterações podem apenas ser propostas, não havendo garantias de que elas serão adotadas pelo usuários.

Quanto à divisão dos trabalhos em aqueles que estudam o computador como mídia e os que estudam a interação dos sistemas computacionais com outros esquemas semióticos, este trabalho tem sua ênfase no segundo aspecto. A interação que será enfocada será do esquema semiótico do software com um esquema semiótico em particular: o registro do usuário.

Na discussão do capítulo anterior a noção de registro apresentada foi: *Registro é a linguagem utilizada pelo usuário.*

Essa noção será agora redefinida, à luz dos tópicos abordados neste capítulo, como: **Registro é um esquema semiótico de uma comunidade de usuários**. Esta noção traz com ela aquela de esquema semiótico: de que é um conjunto de invariantes do uso de signos e que os signos são usadas em um tipo de situação por uma dada comunidade. Estes dados serão importantes para definição da estratégia de software baseada no registro.

O próximo capítulo aborda software como construção de significado e a relação do registro (ou registros) do usuário com a construção e o uso de software.

4. Software como Construção de Significado

> "O conceito chave é usar a linguagem de trabalho como um ponto de partida para inventar novos signos baseados em computador. É mais como fazer uma versão no cinema de um romance do que traduzir um livro de uma língua para outra."
>
> **A Theory of Computer Semiotics** por P. B. Andersen

Um software só é software se assim for interpretado por alguém. Este é o ponto de vista que é colocado neste capítulo.

Através do estabelecimento da perspectiva de mídia para analisar IHC, na seção 4.1, chegamos ao processo de interpretação dos signos da interface (4.2) e daí à relação entre o processo de desenvolvimento do software e esta interpretação (4.3). A interrelação registro do usuário e software é o assunto da seção 4.4. Na seção 4.5 é discutida a abordagem usada neste trabalho para adoção do registro como base para o desenvolvimento de software.

4.1. Sistema Computacional como Interlocutor e como Mídia

IHC pode ser vista de quatro diferentes perspectivas: a de **sistema**, onde usuários são vistos como componentes de entrada de dados para o sistema; a de **parceiro em um diálogo**, onde usuário e sistema são vistos como partes equivalentes em um diálogo; a de **ferramenta**, onde sistemas são vistos como instrumentos manejados pelos usuários; e a de **mídia**, onde sistemas são vistos como meio de comunicação através dos quais mensagens são passadas entre pessoas (Kamersgaard, 1988 op. cit. de Souza, 1993).

Apenas as de parceiro num diálogo e de mídia interessam de um ponto de vista lingüístico e semiótico e elas caracterizam sistemas computacionais como artefatos metacomunicacionais (de Souza, 1993).

Do modelo emprestado da Teoria da Informação temos, de forma simplificada:

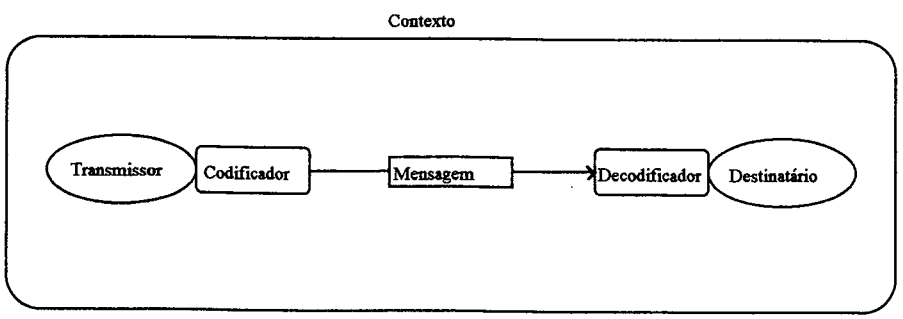

Figura 4.1. Elementos básicos de um processo de comunicação (de Souza, 1993, p. 755)

As perspectivas de parceiro num diálogo (a) e de mídia (b) podem ser vistas, como instâncias do modelo da Figura 4.1., na figura abaixo:

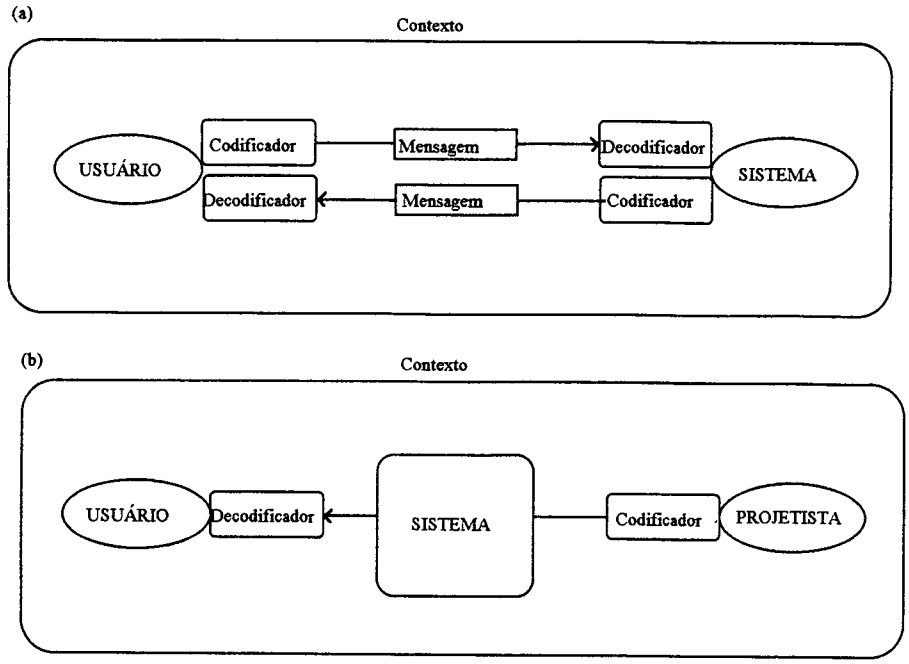

Figura 4.2. Possíveis contextos comunicativos em IHC. (a) Usuário em comunicação com o sistema; (b) Usuário em comunicação com o projetista do sistema (de Souza, 1993, p. 755)

As mensagens que o projetista envia ao usuário (b) são de um tipo especial (*performing messages*) e a *performance* é de um ato comunicacional, só que não mais entre projetista e usuário, mas entre usuário e sistema (a). São encontradas dificuldades devido ao fato de que usuário e projetista não se encontram no mesmo espaço-tempo quando do uso do software, o que torna difícil ou até impossível o *feedback* (de Souza, 1993). Assim, só resta tratar estas questões da comunicação entre projetista e usuário quando da concepção do sistema, tomando-se certas medidas. Já em relação a (a), as dificuldades estão em o usuário comunicar suas intenções ao sistema ('ativando' signos) e interpretar os resultados que lhe são apresentados.

Apesar de a perspectiva de interlocutor poder ser uma boa metáfora, não se pode usá-la como base para um entendimento da IHC, já que não se pode atribuir a faculdade da linguagem aos computadores, pelo menos como os conhecemos hoje. Nos resta a perspectiva de mídia, onde o projetista é visto como delimitador da comunicação e criador de signos que o usuário pode ativar (Andersen, 1990).

A perspectiva de mídia compartilha algumas pressuposições com outras perspectivas, porém difere delas em outros aspectos. Um exemplo é o da orientação-a-objetos: objetos são reinterpretados como signos (propriedades+ações). A idéia de modelo da realidade é abandonada, e com ela a idéia de que o sistema pode ser construído a partir de uma análise de sistema. Programação e projeto são vistos como a criação de signos: "Não haverá representação 'natural' do tópico; ao invés disso, a seleção de informação e a

maneira na qual ela estará estruturada dependerá das tarefas e da cultura dos usuários." (Andersen, 1990, p. 312)

Em resumo, a perspectiva adotada neste trabalho é a de computador como mídia através da qual o projetista se comunica com o usuário. Esta perspectiva leva à reinterpretação do papel do projetista, visto até então como construtor de sistemas, para a de um autor de material simbólico a ser interpretado pelo usuário.

A próxima seção aborda a interpretação do usuário dos signos propostos pelo projetista do software.

4.2 Software e Interpretação

A interpretação que o usuário faz dos signos propostos pelo projetista e que se manifestam no computador é influenciada por vários aspectos, dos quais podemos destacar dois: o código utilizado na codificação por parte do projetista e o contexto onde o sistema é usado, que estabelece as expectativas do usuário. Caso o projetista tenha sucesso, o usuário construirá sua 'metade' do signo (conteúdo) a partir da 'metade' definida pelo projetista (expressão). A figura abaixo ilustra esta visão:

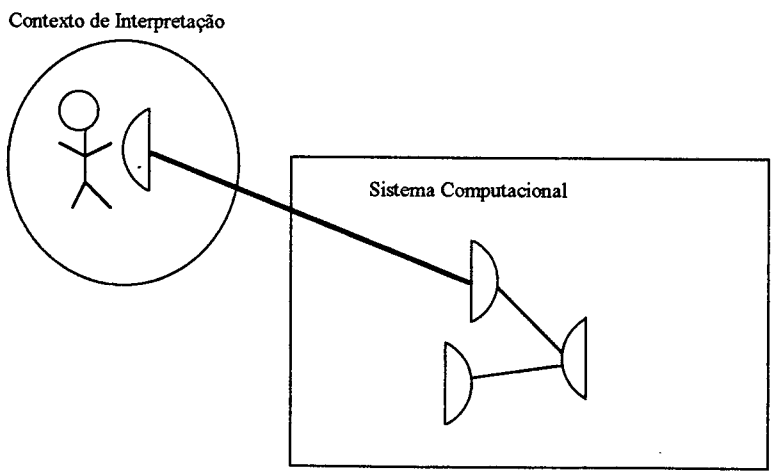

Figura 4.3. A visão básica: o sistema computacional visto como uma coleção de 'metades de signo' (Andersen, 1992, p. 21)

Como exemplo de contexto pode-se tomar o de 'Correio Eletrônico' ou e-mail. Dado este contexto, podemos tentar interpretar a figura abaixo como sendo de um software para correio eletrônico que nos é proposto:

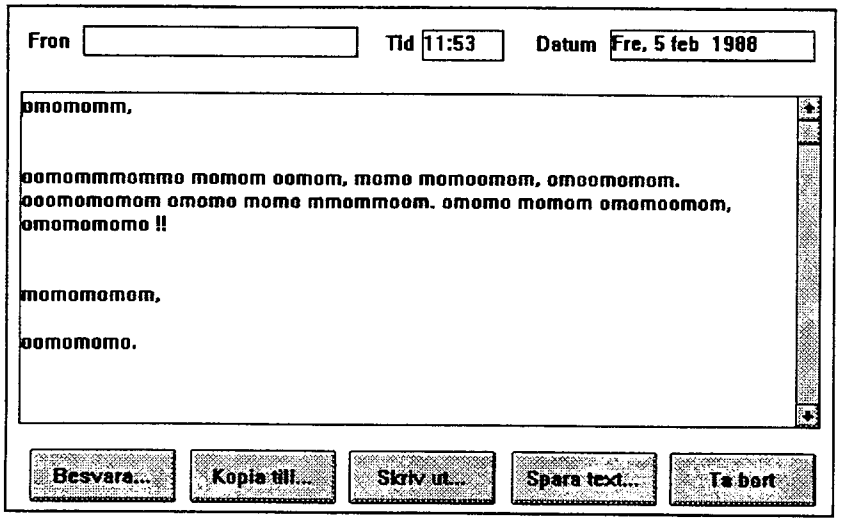

Figura 4.4. Correio Eletrônico em dinamarquês (Andersen, 1990, p. 163)

Dado o contexto, nossas expectativas se modificam e ativamos alguns esquemas para a interpretação do que é apresentado. Embora alguns elementos possam ser identificados, como a hora, data e aquilo que parece ser o texto de uma mensagem, o restante dos signos, como 'Besvara' ou 'Kopia till', não conseguimos interpretar. Falhamos então em construir a nossa 'metade' do signo porque foi usado um código que nos é estranho na codificação, resultando em uma decodificação problemática. Caso o código utilizado fosse o do português, teríamos um resultado como o abaixo:

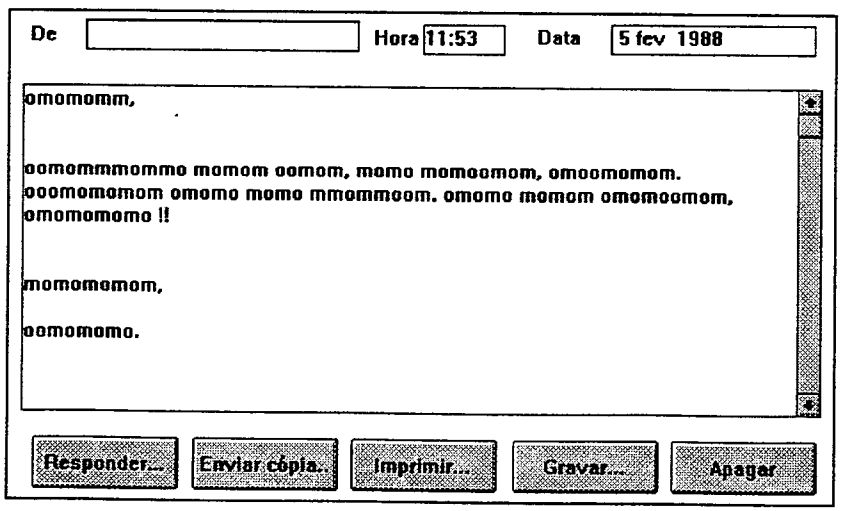

Figura 4.5. Correio Eletrônico em português

Apesar de os signos (expressão) manifestados na execução do programa em dinamarquês terem sido concebidos com a intenção de que aquele fosse um software de correio eletrônico, isto não ocorreu. É importante, assim, perceber a diferença entre programa de computador e software. Dentro desta discussão de software como interpretação, a seguinte definição é apropriada:

> Programa de computador é um conjunto de instruções executáveis por um computador.

A noção de software, por outro lado, não é a de um produto, de um objeto, mas de um processo:

> Software é quando um programa de computador é interpretado como software de algo por alguém.

Esta distinção é proposta aqui para enfatizar a importância da interpretação no fenômeno software e, por consequência, seu caráter semiótico. Tudo aquilo que concorre para que um programa de computador seja interpretado como software é importante no processo: a interface em si, *help on-line*, manuais, treinamento, e outros. Um destes fatores, de enorme importância, é o da Análise de Requisitos, onde são especificadas que condições o sistema computacional deverá satisfazer. Esta atividade ocorre no início da concepção do sistema e tem influência em todo o processo de desenvolvimento. Este tema será retomado na seção seguinte, 4.3. Análise de Requisitos e Interface.

Tomemos agora o exemplo de um programa de computador que nos é apresentado como um software para controle de conta-corrente. Caso exista um botão rotulado de 'Depósito...', nossa expectativa será de que a ativação deste leve a solicitação de dados sobre o depósito, incluindo seu valor e que este seja adicionado ao saldo atual e resulte em um novo saldo. Se o saldo continuar o mesmo, ou então for subtraído, não ocorrerá aquilo que previamos. Uma interpretação possível é que a expressão está de acordo com as expectativas, pois 'Depósito' faz parte do campo semântico associado à conta-corrente. Porém esta expressão está articulando um outro conteúdo, de um domínio onde 'depositar' não está associado a 'incrementar o saldo'. Assim, não basta que a forma da expressão seja a esperada, mas que esta articule da forma esperada o tópico do sistema. Isto também leva a considerar a funcionalidade, e não só aquilo usualmente descrito como 'interface', como contribuindo para a interpretação (Andersen, 1990).

O problema em ambos os casos (do correio eletrônico e da conta-corrente) é que a forma (da expressão e do conteúdo) não corresponde àquela de quem vai usar o software. Onde encontrar esta forma ? Como já foi visto forma é inerente aos esquemas semióticos, onde são registrados os invariantes. Assim, um esquema semiótico que articule estes dois planos é o lugar onde se deve procurar ajuda. Este lugar geralmente existe como um esquema lingüístico denominado de registro.

Nas próximas seções serão discutidas as relações entre o processo de desenvolvimento de software e o registro.

4.3 Análise de Requisitos e Interface

O programa é o componente do sistema que vai descrever a substância da expressão dos signos baseados em computador (Andersen, 1990): "os processos do sistema são substâncias que podem se transformar em expressões em um processo interpretativo que simultaneamente estabelece seu conteúdo." (Andersen, 1990, p. 129)

Anteriormente, quando da colocação da perspectiva de mídia para análise dos sistemas computacionais, foi dito que as mensagens (os sistemas) quando do uso dos programas, são unidirecionais, do projetista ao usuário, ambos em tempo e espaço possivelmente diferentes, o que torna o *feedback* praticamente impossível. Assim, uma maneira de tratar as dificuldades neste processo comunicacional é quando da concepção do software, envolvendo o usuário no processo de desenvolvimento.

A atividade que serve de base para a construção de programas é a Análise de Requisitos. Ela costuma ser descrita em Engenharia de Software como a primeira atividade do ciclo tradicional de desenvolvimento, e dela resulta um documento, a descrição dos requisitos, a partir do qual um sistema é especificado. É justamente nesta atividade, onde o usuário está envolvido, que são relatados problemas gravíssimos de comunicação: "a maioria dos problemas enfrentados pelos analistas diz respeito à comunicação com os usuários. ... Embora eles (analistas) saibam que a terminologia do método de desenvolvimento de sistemas não seja apropriada como uma linguagem em comum para falar com os usuários, eles têm dificuldade em achar um meio de comunicação mutuamente aceito. ... A comunicação analista/usuário não é contemplada por nenhum método de desenvolvimento de sistemas atual." (Browne, 1992, p. 46)

Embora a comunicação tenha um papel reconhecidamente importante também no processo de construção de programas, e que, de um ponto de vista semiótico, este seja essencial para que o programa seja interpretado como software, este aspecto vem recebendo tratamento falho dos métodos de desenvolvimento de sistemas, concebidos na Ciência da Computação, onde um discurso humanista somente agora começa a ter implicações de ordem prática.

Nas seções seguintes será elaborada a relação entre registro lingüístico e software e discutido que registro pode ser utilizado para tornar mais efetiva a comunicação analista/usuário tanto na construção como no uso de programas.

4.4. Registro e Software (Andersen, 1992)

> "Um *registro* é a linguagem usada *em* um determinado tipo de situação com o propósito de suportar ou mudar suas atividades." (Andersen, 1990, pg. 54)

Um usuário em potencial deve ter internalizado vários registros, usados na situação de trabalho e em outros tipos de situação. Assumindo-se que a construção e o uso do software são atividades suportadas por e realizadas através de comunicação, é necessário que exista um sistema de significação subjacente. Uma proposta de sistema de significação já existente e que poderia ser usado é o registro, de maneira a se estruturar a forma da expressão e do conteúdo da interface.

A forma do conteúdo dos signos da interface pode se manifestar em duas substâncias, já que eles podem se referir a processos e objetos computacionais ou a objetos e processos na área a que a aplicação se destina. Quando a primeira situação ocorre temos o *significado formal* dominante (registro usado como metáfora), e a segunda é denominada de *significado real* dominante (registro usado com seus significados originais, se referindo a objetos externos ao computador).

A estratégia utilizada dependerá do tipo de software em questão:

- **tarefa já é executada pelo trabalhador**, portanto provavelmente já existe um registro usado para designar objetos e processos da tarefa. Neste caso usaremos o significado real dominante.

- **tarefa ainda não executada pelo trabalhador,** e assim não há um registro para se discorrer sobre ela. Aqui um registro já conhecido pelo usuário e com forma do conteúdo semelhante será usado para que, ao menos inicialmente, tenhamos uma metáfora que acelere a aprendizagem e a eficiência do projetista do software em comunicar seus propósitos.

Como exemplos podemos ter um sistema de controle de estoque e um sistema de arquivos (cujo próprio nome é uma metáfora, hoje possivelmente morta para a maioria).

No controle de estoque deve-se procurar respeitar o registro usado pelo possíveis usuários, que já desempenham as tarefas relativas ao controle de estoque, para denominar e organizar os objetos e processos do software. Portanto, serão usados termos tais como 'item em estoque', 'entrada', 'saída', 'estoque mínimo' para designar elementos que possuem uma contraparte externa ao computador. A forma do conteúdo destes signos será projetada também para respeitar as relações que estes elementos possuem na atividade de controle de estoque.

No sistema de arquivos o que se procurou foi fazer uma metáfora com algo que a maioria das pessoas conhece. Assim, estabeleceu-se a idéia de um arquivo onde

informações podem ser armazenadas. Os arquivos devem ser abertos para ser usados e fechados quando não mais necessários. Em alguns sistemas pode-se jogar um arquivo na lata de lixo quando sua utilidade cessa.

A abordagem do significado real dominante foi a usada nos estudos de casos que serão descritos no capítulo 7. Porém, uma estratégia que se proponha a trabalhar com registro pode servir tanto para sistemas de significado real quanto de formal dominante. Isto dependerá apenas do registro escolhido. Andersen (1990) traz um exemplo de uma rede de computadores em que o significado formal de um registro para discorrer sobre encanamentos, usando termos tais como 'cano de A a B' e 'abrir torneira', foi usado para designar elementos e operações da rede.

O que é importante notar nas duas abordagens é que o uso do registro deve tornar o sistema mais fácil de aprender e de usar. A próxima seção discute dois tipos de registros existentes referentes a uma mesma situação (registro usado na situação e para discorrer sobre ela) e qual destes tipos de registro deve ser usado na construção de programas.

4.5. Discussão: Registro de Trabalho e Registro sobre o trabalho

"... trabalho real não é sempre enfocado nos objetos de trabalho, mas em certas situações os trabalhadores precisam sair fora do trabalho e enfocar suas pré-condições técnicas e organizacionais, e no caso particular do trabalho informatizado significa concentrar nas propriedades do próprio sistema - eles precisam desvendar o significante."(Andersen, 1990, p. 302). Duas situações em que este mudança de foco é necessária: *mystery solving* (onde os trabalhadores tentam interpretar um comportamento do sistema) e *forecasting* (inventam métodos e tentam imaginar se o sistema vai viabilizá-los ou não). Além disso, na abordagem deste trabalho, este tipo de comportamento é preciso quando do projeto do sistema, utilizando-se para tal registro sobre o trabalho.

No artigo *Work Language Analysis and the Naming Problem*, Katzenberg & Piela (1993) chamam a atenção, baseados em Holmqvist & Andersen (1987), de que a variedade apropriada para a interface é linguagem de trabalho e não sobre o trabalho. Isto parece estar em conflito com a presente abordagem, no entanto há uma reconciliação ao dizerem que a principal fonte de dados são as explicações a pessoas de outros grupos (e.g. entre novatos e experientes). Andersen (1990) na parte III, em que trata da ligação entre trabalho e projeto de sistemas, também coloca entrevistas e bate-papos como fontes de informação sobre o trabalho.

Além disso, Katzenberg & Piela (1993) estão tratando de nomear a interface de um sistema que deve funcionar **em** uma situação de trabalho. Não é **na** situação de trabalho que o software, e seus requisitos, serão discutidos, mas sim quando o trabalhador numa atitude reflexiva, fora da situação de trabalho (e usando portanto linguagem **sobre** o trabalho), pensar sua situação de trabalho, verificar o que ocorre e que mudanças poderiam ser suportadas por um sistema computacional. Aí entra o analista de requisitos, que, por sua vez, inteirado da linguagem **sobre** trabalho, discutirá o que o software deverá

suportar e implementará um sistema computacional que suporte tais mudanças (que porções do sistema de informação serão suportadas pelo sistema computacional).

Andersen no epílogo de seu trabalho seminal, coloca a necessidade de se desenvolverem métodos de projeto baseados na semiótica: "Tais métodos podem ver a atividade de projetar como **conversações** cujos participantes **inventam novos signos baseados em computador**..." (Andersen, 1990, p. 403). Essa conclusão define o projeto como uma conversação, ou seja, uma atividade em que a comunicação é importante e na qual um código deve ser usado. No entanto, um sistema de controle de estoque de peças de uma oficina mecânica não poderia ser concebido com produção de enunciados tais como 'Troque o carburador daquele carro'.

Assim, precisamos ter conhecimento de um esquema semiótico que sirva para o projeto dos sistemas, atividade que se dá fora da situação de trabalho, criticando-a e propondo um sistema que suporte partes deste trabalho. Nesta atividade serão inventados os signos (propostas de) da interface, que podem estar relacionados com o registro sobre o trabalho, na forma de extensões ou combinações. No entanto, um processo de prototipação é necessário para se avaliar como estes signos interagem com o registro de trabalho.

Resumindo, duas variedades de registro interagem com o software: na concepção o *registro sobre o trabalho* é usado nas discussões que vão inventar os signos baseados no computador; e no uso onde estas propostas de signos vão interagir com o *registro de trabalho*, usado na situação de trabalho. É esta última que vai definir o sucesso dos signos propostos ao serem adotados ou não pelos usuários.

A variedade de linguagem, ou registro, adotado na presente abordagem é o **registro sobre o trabalho**. O LAL (Léxico Ampliado da Linguagem) e o DOM (Domain Object Model) são duas estruturas onde serão representados dados sobre este registro. Os capítulos seguintes tratarão destas e de seu uso em dois estudos de caso, discutirão alguns resultados e estabelecerão a continuidade deste trabalho.

5. Léxico Ampliado da Linguagem: Um Modelo de Representação do Registro

O Léxico Ampliado da Linguagem (LAL) é uma estrutura proposta pelo Prof. Júlio Leite, da Pontifícia Universidade Católica do Rio de Janeiro. Ela foi concebida dentro do contexto da pesquisa em Análise de Domínio, uma área da Engenharia de Software que se preocupa com questões como a reusabilidade e a comunicação analista/usuário.

O LAL é uma ferramenta para a coleta e representação de dados sobre o registro do usuário. Sua organização é semelhante a de um mini-dicionário e servirá para suportar a comunicação na atividade da Análise de Requisitos.

Este capítulo procura situar o contexto onde esta estrutura surgiu, como construí-la, qual a sua reinterpretação neste trabalho e alterações feitas à proposta original. O LAL é um dos instrumentos da estratégia de software centrada no registro explorada neste trabalho.

5.1. Análise de Domínio

No processo tradicional de desenvolvimento de software, dividido em fases, o resultado de uma fase é usado como ponto de partida para a fase seguinte. Isto acontece com todas as fases com exceção da Análise de Requisitos, que, sendo a primeira do ciclo, não tem suporte de nenhuma outra fase. Esta falta de suporte leva a dificuldades nesta fase, sendo que uma das mais notórias, como já citado, é a dificuldade de comunicação entre analista e usuário.

Tendências recentes têm apontado para o uso da Análise de Domínio, numa fase anterior à Análise de Requisitos de forma a apoiá-la (Prieto-Diaz, 1987). Esta se dá pela construção de um modelo do domínio onde se dará o desenvolvimento de aplicações, procurando elicitar a linguagem característica do domínio. Esta será então usada para descrever os requisitos da aplicação (e, na abordagem deste trabalho, na comunicação analista/usuário) e assim se dá um tipo poderoso de reuso, o de análise.

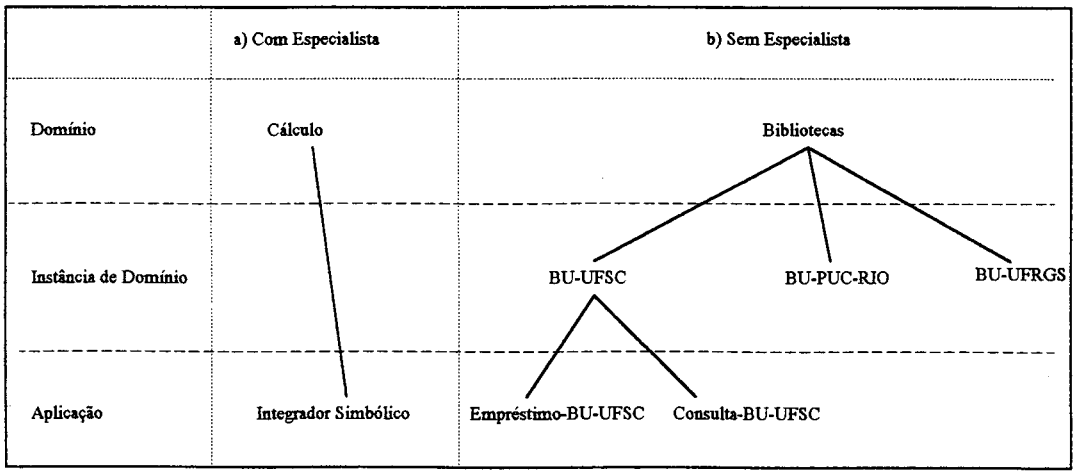

Figura 5.1. Abordagens à Análise de Domínio

Para levantar a linguagem característica geralmente adota-se o procedimento de consultar um especialista. Nos domínios bem estabelecidos e formalizados isto é possível, sendo as aplicações descritas diretamente usando conhecimento representado sobre o domínio (Figura 5.1, item a). Porém, existem alguns domínios onde são raros ou inexistentes os especialistas. Tome-se por exemplo domínios bastante informais, tais como uma banca de revistas ou um restaurante. Nesses casos não existem especialistas no domínio, mas os próprios participantes de uma banca de revistas ou de um restaurante, tais como um vendedor ou um caixa. Cada ocorrência de um domínio é chamada de instância do domínio. Temos, por exemplo, o domínio Bibliotecas e a instância deste domínio BU-UFSC (Figura 5.1, item b).

Precisamos explicitar e validar técnicas para levantar o conhecimento junto a cada um destes sujeitos. O LAL é um instrumento concebido para implementar uma 'estratégia redutora' ao problema de construção de linguagens de domínio.

5.2. LAL

A proposta de Neighbors (1984) de construção de programas através de componentes reutilizáveis exige que se faça uma análise de domínio para construir uma linguagem específica ao domínio. Esta linguagem é então usada para especificar o software, e esta especificação é, por sua vez, transformada em programas executáveis. Este paradigma ficou conhecido como paradigma Draco.

Leite (1989) propôs a Linguagem da Aplicação como uma maneira de se "ganhar mais experiência na aquisição e produção de linguagens orientadas a problemas, antes de generalizar os resultados para as linguagens de domínio" (Leite & Franco, 1990, p. 135).

O Léxico Ampliado da Linguagem é uma estrutura associada a Linguagem da Aplicação que serve para representar os termos pertinentes a esta linguagem com suas respectivas definições.

5.2.1 Estrutura

Cada entrada do LAL é composta de três partes: o *nome da entrada* (e sinônimos), as *noções* (definição) e os *impactos* (efeitos do uso ou da ocorrência).

As noções e impactos devem obedecer dois princípios: *circularidade* e *vocabulário mínimo*. A circularidade diz que os termos da linguagem devem ser descritos usando-se termos da própria linguagem e o vocabulário mínimo diz que o uso de termos externos à linguagem deve ser minimizado.

Abaixo, parte do LAL de um restaurante (Franco & Leite, 1992, p. 206-207):

CLIENTE/FREGUÊS

Noções:
1. pessoa que consome OPÇõES do RESTAURANTE

Impactos:
1. realiza as ações: SENTAR-NA-MESA, FAZER-O-PEDIDO, TROCAR-DE-MESA, PEDIR-A-CONTA, PAGAR-A-CONTA

FAZER-O-PEDIDO/FAZ-O-PEDIDO

Noções:
1. ação realizada pelo CLIENTE
2. o CLIENTE escolhe uma OPÇÃO
3. o CLIENTE informa o seu PEDIDO ao GARÇON

Impactos:
1. o GARÇON anota o PEDIDO do CLIENTE na COMANDA
2. o GARÇON deve JOGAR-A-COMANDA

ABRIR-A-MESA/ABRE-A-MESA/ABRIU-A-MESA

Noções:
1. tarefa realizada pelo CAIXA
2. acontece quando o CLIENTE SENTA-NA-MESA e FAZ-O-PEDIDO
3. o CAIXA verifica se a MESA-NÃO-ESTÁ-ABERTA
4. o CAIXA RECEBE-A-COMANDA e COLOCA-A-COMANDA-NO-ESCANINHO

Impactos:
1. a MESA-ESTÁ-ABERTA
2. se a MESA-ESTÁ-ABERTA então o CAIXA não pode ABRIR-A-MESA

5.2.2. Elicitação

O processo de elicitação do LAL se compõe de basicamente quatro atividades (Franco & Leite, 1992):

- *Identificação das fontes de informação*: procura-se identificar documentos existentes e quais são as pessoas mais envolvidas com a aplicação.

- *Identificação dos termos*: através da observação, de entrevistas e da leitura de documentos. O objetivo é elaborar uma lista de termos.

- *Identificação da semântica*: se constitui da elaboração da descrição (noções e impactos) para cada termo da lista, seguindo os princípios da circularidade e do vocabulário mínimo.

- *Validação*: validação informal junto aos informantes e validação estrutural do léxico, verificando a falta de termos pela quebra da circularidade.

5.2.3 Utilização

Terminada a validação, o LAL está pronto para ser usado. O grupo de pesquisas da PUC-Rio tem investigado a derivação de um modelo conceitual mais 'semanticamente rico', para uso em Engenharia de Requisitos. O modelo adotado por eles foi o KAOS (Lamsweerde, 1989) e um conjunto de heurísticas é usado na derivação, em um processo de casamento de padrões (Franco & Leite, 1992).

5.3 Discussão

A abordagem de Leite compartilha com este trabalho a inspiração semiótica e a preocupação antropológica de caracterizar uma cultura através de sua linguagem. Nos capítulos anteriores foi buscada uma caracterização de software que enfatiza a interação deste com outros esquemas semióticos, sendo importante para o projeto de sistemas o reconhecimento destes esquemas. O esquema escolhido para ser trabalhado foi o do registro sobre o trabalho onde o sistema computacional será usado e o LAL é o instrumento de elicitação e representação deste conhecimento.

Eco (1979) em uma discussão sobre o interpretante da teoria de Peirce, destaca a *semiose ilimitada*. Esta acontece porque só podemos ter acesso aos signos através de um processo de tradução de um signo em outro (o interpretante). O LAL implementa esta noção através das noções e impactos, que definem um termo da linguagem através de outros. Os princípios da circularidade e do vocabulário mínimo permitem lidar com a semiose ilimitada. A circularidade é uma propriedade dos sistemas de significação desenvolvidos pelo homem em sua história cultural (Eco, 1979), e este princípio aplicado ao LAL faz com que nenhum dos termos concernentes à linguagem (registro) específica fique sem definição. O princípio de vocabulário mínimo transfere o problema de definição dos termos não pertinentes ao registro a um esquema semiótico mais genérico.

Já a atividade de determinar que termos são pertinentes ao registro pode ser concebida como uma aplicação do Teste da Comutação. Aqueles termos que comutam no esquema específico, ou registro, diferentemente do esquema mais genérico ou não comutam neste último são pertinentes ao registro. Tome-se, por exemplo, o termo 'bater na posição' (Leite & Franco, 1990). Este termo comutaria, por exemplo, com 'bater fora da posição' em linguagem coloquial, articulando um outro conteúdo, e teria como

possíveis interpretantes 'martelar na posição' ou 'golpear na posição'. Já no domínio de custódia de ações este termo tem como interpretante 'crédito de títulos na conta mãe', e não comuta com 'bater fora da posição'. Podemos perceber que o mesmo termo comuta de forma diferente no domínio específico de custódia de ações, do que o faz em um contexto mais geral e que em para cada um dos contextos os interpretantes possíveis mudam. Assim, através da aplicação do Teste da Comutação, podemos determinar se um termo é pertinente ou não ao registro. Adicionalmente, este teste pode contribuir para a determinação dos termos sinônimos: são termos que comutam da mesma forma em relação aos demais, a comutação de um pelo outro não articula outra porção da forma do conteúdo, e eles podem ser interpretantes uns dos outros.

Os termos que comutam da mesma forma no contexto específico e no genérico podem ser usados no vocabulário mínimo. O termo 'ação', por exemplo, usado na descrição da entrada FAZER-O-PEDIDO (v. pg. 29), comuta da mesma forma no contexto específico e no genérico, não tendo um significado particular dentro do domínio dos restaurantes. Desta forma pode ser interpretado de forma apropriada por quem não tem conhecimentos específicos ao domínio (o analista, por exemplo).

O problema de delimitação do escopo do domínio (ou da instância do domínio) também pode ser tratado com uma abordagem semiótica. Este, que é um dos problemas de difícil tratamento pelo Análise de Domínio (Prieto-Diaz, 1987), pode ser abordado através do teste da comutação: se um termo comuta da mesma maneira dentro e fora do domínio específico, então não pertence a este domínio, mas a algum domínio mais genérico. E se um termo comuta dentro do domínio de maneira diferente que fora deste, ou não comuta, então pertence ao escopo[3].

O objetivo das colocações acima é duplo: mostrar que o LAL é apropriado como representação de dados do registro; e estabelecer uma ligação entre conceitos de semiótica e trabalhos que vêm sendo realizados na pesquisa em informática. A seguir é feito o confronto da abordagem de LAL usada neste trabalho e a proposta original de Leite.

Embora compartilhando vários aspectos, a presente abordagem difere da de Leite nos seguintes pontos:

- *registro de trabalho e sobre o trabalho*: Leite enfoca o registro de trabalho enquanto que a variedade aqui adotada é a do registro sobre o trabalho (ver seção 4.5. Discussão: Registro de Trabalho e Registro sobre o trabalho).

- *observação ou entrevistas*: para o levantamento dos dados sobre o registro Leite usa observação e entrevistas, dada sua opção de registro de trabalho, cujo uso pode

[3]Na verdade isto geraria uma hierarquia de domínio-subdomínio. Por exemplo, os termos PROFESSOR, DAR-AULA e ALUNO comutam em Escola (domínio) e em Escola de Idiomas (subdomínio) de forma diferente, mas relacionada, onde os termos do domínio são mais genéricos em suas definições que os termos do subdomínio.

ser observado na atividade de trabalho. As entrevistas são usadas para complementar a observação e para validação. Na presente abordagem são utilizadas apenas entrevistas, para manter um ponto de vista sempre o mais próximo daquele do usuário.

- *número e tipo das categorias dos termos*: Leite trabalha com um número propositadamente pequeno de categorias (quatro) para classificação dos termos do LAL. Estas categorias de inspiração sintática são: sujeito, verbo, objeto e estado. Para cada uma destas categorias foram definidos noções e impactos (Leite, 1993):

Categoria	Noções	Estados
Sujeito	Quem ele é	Que ações executa
Verbo	Quem executa a ação Quando ele acontece Quais os procedimentos envolvidos	Restrições quanto a ocorrência da ação Ações decorrentes Situações causadas
Objeto	Identificação Relacionamentos com outros objetos	Ações que podem ser aplicadas
Estado	O que é Que ações levaram a ele	Que estados ou ações podem decorrer

Figura 5.2. Categorias do LAL segundo a proposta original de Leite

Estas categorias foram consideradas, durante os estudos de caso, insuficientes e pouco descritivas. O termo BANCA, por exemplo, só podia ser categorizado como **objeto**, enquanto que nenhuma ação era aplicada a ele (ver tabela acima), mas sim ações aconteciam na BANCA, objetos eram transportados de e para lá.

A teoria de Gramática de Casos (Fillmore, 1968) trouxe inspiração para tratar este problema. Esta teoria propõe que se trabalhe não com categorias sintáticas, mas sim com categorias que reflitam o significado dos termos. Assim, o termo BANCA, dentro desta teoria, não seria classificado como objeto, mas como **lugar**, **origem** e **destino** de ações. Isto resulta porque dentro desta teoria os constituintes de uma sentença caem em um pequeno número de tipos, definidos pela suas relações com a ação denotada pelo verbo. Estes tipos usualmente são:

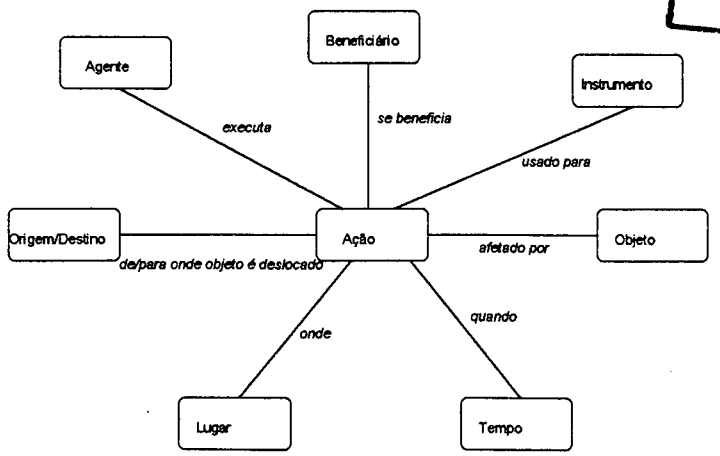

Figura 5.3. Casos e relações com a ação

A partir destes tipos foram definidas um novo esquema de classificação para os termos do LAL, sendo mantida a categoria estado:

Categoria	**Noções**	**Impactos**
Agente	Definir o agente	Ações que executa
Objeto	Definir o objeto	Ações que podem ser aplicadas ao objeto
Instrumento	Definir o instrumento	Como ele é usado na execução da ação
Beneficiário	Definir o beneficiário	Como ele é afetado por uma ação Que ação o afeta
Lugar	Definir o lugar	Ações que ocorrem no lugar
Tempo	Definir o tempo	Ações que ocorrem neste tempo
Origem	Definir a origem	Ações que deslocam algo da origem
Destino	Definir o destino	Ações que deslocam algo para o destino
Ação	Quem executa a ação Quando a ação é executada Procedimentos envolvidos	Consequências: . Ações que deverão ocorrer . Situações decorrentes Situações que impedem sua ocorrência
Estado	Definir o estado (Ações causadoras e objetos relacionados ao estado)	Consequências . Situações causadas . Ações causadas

Figura 5.4. Categorias do LAL inspiradas na Teoria das Gramáticas de Caso

A noção de caso, como relação do termo com a ação, está refletida nos impactos. Um termo terá, ao menos, tantos impactos quantas forem suas relações com a ação. Assim, o termo BANCA terá ao menos três impactos, já que seus casos são três (lugar, origem e destino).

Foi tratado neste capítulo de um instrumento para coleta e registro de dados sobre o registro do usuário sobre o trabalho. Um breve histórico foi traçado, conceitos foram reinterpretados dentro da semiótica e a abordagem usada nos estudos de caso foi diferenciada em relação à proposta original.

No capítulo seguinte o DOM, um modelo de representação de dados do registro será definido. Este modelo pretende tratar do aspecto de reusabilidade e de formalização crescente do registro para suporte à construção de programas. Ele parte dos dados constantes do LAL e os organiza em um modelo que servirá, dentre outros propósitos, para dar suporte à Análise de Requisitos.

6. DOM: Organizando e Reusando o Registro

A construção de representações em um computador é uma atividade que requer a tradução da representação para uma linguagem artificial, e que possibilite um tratamento formal (manipular cadeias de símbolos, de acordo com regras pré-definidas, sem ser necessário conhecer o significado dos símbolos).

Este capítulo descreve o desenvolvimento atual de um modelo de representação de dados sobre o registro em computador. Este modelo é denominado Domain Object Model (DOM). Originalmente proposto por Faust (1992), e inspirado nas facilidades representacionais do *Knowledge Representation and Inference SYStem*, KRISYS (Mattos, 1989), seu objetivo é promover a reusabilidade das informações sobre o registro.

Nas próximas seções serão descritos os objetivos, a atual arquitetura do modelo, a tradução LAL-DOM, as formas de reuso previstas e como este se encaixa na perspectiva semiótica adotada neste trabalho.

6.1. Objetivos

DOM foi proposto originalmente como "um sistema de software que implementará um espaço para representação de elementos do Léxico Ampliado da Linguagem" (Faust, 1992). Este espaço, no entanto, não organiza os termos da mesma forma que o LAL o faz, com descrições feitas em linguagem natural, mas sim usando um conjunto pré-definido de objetos, *slots* e aspectos (ver seção 6.2. Arquitetura).

Para promovermos a reusabilidade de informações precisamos ser capazes de comparar um novo caso com aqueles que já conhecemos, para saber que informações podem ser reusadas. Se um computador for utilizado para auxiliar nesta tarefa, é preciso representar as informações em um modelo manipulável pelo computador. A maneira mais fácil de fazer isso é utilizar um modelo tratável pelo computador, baseado em um conjunto de elementos manipuláveis, sem levar em conta seus significados, mas de forma válida e interpretável por quem conhece o modelo.

Um modelo deste tipo também está mais perto de uma linguagem de programação, que será um passo necessário na construção de programas. No caso do DOM, que é um modelo orientado-a-objetos, o foco serão as linguagens de programação também desenvolvidas neste paradigma.

De posse de um modelo com estas características para representar os dados coletados no LAL sobre o registro dos usuários, os seguintes objetivos foram estabelecidos:

- Eliminar possíveis falhas na elicitação do LAL, já que os fatos tem que ser explicitados, o que torna mais fácil a localização de incoerências e de omissões;

- Promover a reusabilidade das informações do registro sobre o trabalho, possibilitando através da formalização saber o que comparar para reusar;

- Servir de base para construção de software, estabelecendo um *framework* de classes para desenvolvimento de aplicações naquela instância de domínio.

Estes objetivos serão reexaminados quando da descrição dos estudos de caso, no capítulo seguinte, quanto a sua exeqüibilidade.

6.2. Arquitetura

Definir uma linguagem artificial consiste em propor uma forma da expressão e a associação desta a uma forma do conteúdo, para um uso específico.

Para propor a forma da expressão, precisa-se trabalhar com algum tipo de gramática. Esta gramática (meta-linguagem) vai definir que elementos um 'texto' válido pode conter e como estes elementos podem ser combinados.

A associação a uma forma do conteúdo pode ser feita através de um texto em outra linguagem já internalizada pelo usuário da linguagem artificial sendo definida, levando-se em conta o contexto. Esta vai permitir que pessoas interpretem o modelo, associando-lhe conteúdo.

Os elementos básicos do DOM são: **objetos**, **slots** e **aspectos**. Slots estão associados a um objeto, assim como aspectos estão associados a slots. A estrutura resultante pode ser visualizada abaixo:

```
Objeto1
  slot1
  slot2
    aspecto1
    aspecto2
  slot3
    aspecto4
...
```

Enquanto objetos podem ser de apenas um tipo, slots e aspectos são divididos em vários grupos. Os slots podem ser de dois tipos: **atributos** (definem uma propriedade do objeto; para uma pessoa poderiam ser idade e peso, por exemplo) e **métodos** (são ações executadas pelo objeto; para uma pessoa poderiam ser caminhar, trabalhar e dormir, por exemplo).

Os aspectos têm sua classificação determinada pelo tipo de slot a que estão associados, por exemplo: `pré-condições`, `pós-condições`, `Lugar` (para um método); `valor` (para um atributo). Aspectos podem ser simples ou multivalorados.

Abaixo um exemplo de um objeto do DOM/Banca de Revistas:

```
FORNECEDOR
superClasseDe
   valor (FORNECEDOR-DE-REVISTAS-E-LIVROS
        FORNECEDOR-DE-JORNAIS)
FAZER-O-CADASTRO
   argumentos  (BANCA)
```

O objeto sendo descrito é `FORNECEDOR`. Ele possui dois slots: `superClasseDe` (atributo) e `FAZER-O-CADASTRO` (método). O atributo `superClasseDe` possui um aspecto, `valor` , que é multivalorado; e o método `FAZER-O-CADASTRO` possui um aspecto, `argumentos`, o qual possui apenas um valor. O atributo `superClasseDe` é de um tipo especial chamado de atributo estrutural, através dos quais são representados relacionamentos entre objetos. Estes relacionamentos e os atributos estruturais associados serão descritos a seguir.

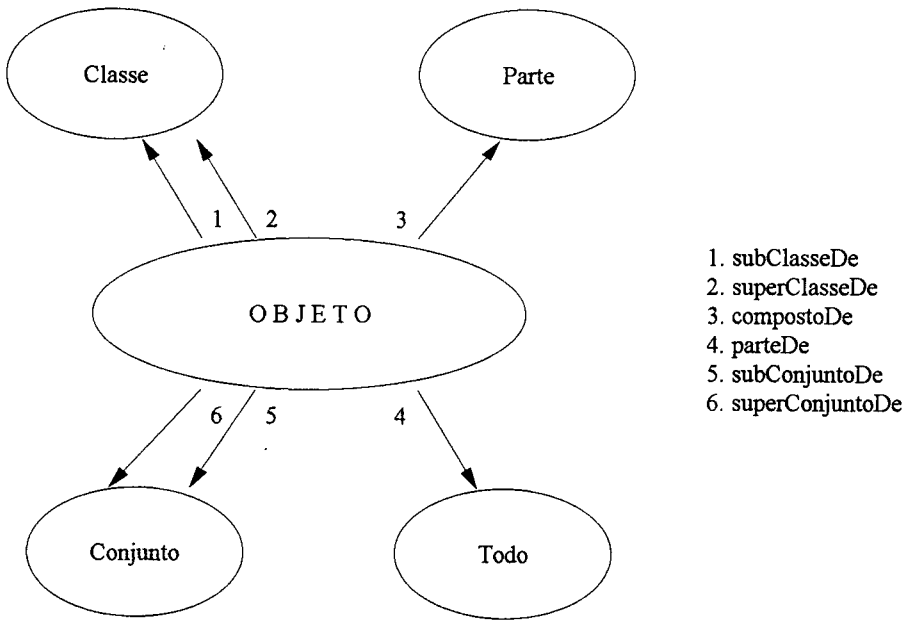

Figura 6.1. Relacionamentos de abstração no DOM, adaptado de (Mattos, 1989, p. 125)

"Do ponto de vista dos conceitos de abstração, cada objeto é ou simples (i.e., definido em si mesmo) ou composto (i.e., definido como uma abstração de outros objetos)" (Mattos, 1989, p. 108). O registro do usuário pode muitas vezes conter termos cujos referentes são objetos compostos, como `FORNECEDOR`, na Banca de Revistas, que é composto a partir de `FORNECEDOR-DE-JORNAIS` e `FORNECEDOR-DE-REVISTAS-E-LIVROS`. Portanto, é interessante provermos meios de representar estes relacionamentos no DOM.

Na figura 6.1. estão listados os relacionamentos suportados pelo DOM. Estes podem ser divididos em três grupos, de acordo com a abstração com que se relacionam (Mattos, 1989):

- `subClasseDe/superClasseDe`: estão relacionadas à abstração de **Classificação/Generalização**. Classificação é a relação entre objetos simples e um objeto composto, a classe, que define propriedades compartilhadas pelos objetos simples. O termo `FORNECEDOR-DE-JORNAIS`, se refere a uma classe de instâncias concretas de fornecedores de jornais. Generalização é um relacionamento de múltiplos níveis entre objetos compostos (classes), onde a superclasse é mais genérica que as subclasses: `FORNECEDOR` é a superclasse de `FORNECEDOR-DE-JORNAIS` e de `FORNECEDOR-DE-REVISTAS-E-LIVROS`.

- `compostoDe/parteDe`: estão relacionadas à abstração de **Agregação**. Esta abstração trata objetos como uma composição de outros objetos. Um exemplo usual é o de um automóvel que é tratado como a agregação dos objetos motor, rodas e chassi. Termos do registro freqüentemente se referem a este tipo de objeto, por exemplo, `MATERIAL-DE-APOIO` no estudo de caso Escola de Idiomas é um objeto composto de `SCHEDULE` e `VISUAL-AIDS`.

- `subConjuntoDe/superConjuntoDe`: estão relacionadas à abstração **Associação**. A abstração de Associação define objetos compostos denominados de conjuntos. A cada conjunto está associado um predicado de pertinência, que determina que objetos simples comporão o conjunto. Um objeto se torna subconjunto de outros restringindo o predicado de pertinência. No estudo piloto Banca de Revistas do Marcelo, um dos termos elicitados foi `PRODUTOS-À-VENDA` cujo predicado de pertinência é que o estado do `PRODUTO` deve ser igual a `DISPONÍVEL-PARA-VENDA`.

Para representar uma relação de abstração entre dois objetos, associamos a cada um dos termos uma ocorrência do par, de forma apropriada. Por exemplo, o par `subClasseDe/superClasseDe`:

```
FORNECEDOR
superClasseDe
   valor (FORNECEDOR-DE-REVISTAS-E-LIVROS
      FORNECEDOR-DE-JORNAIS)
...

FORNECEDOR-DE-JORNAIS
subClasseDe
   valor (FORNECEDOR)
...

FORNECEDOR-DE-REVISTAS-E-LIVROS
subClasseDe
   valor (FORNECEDOR)
...
```

A descrição feita até aqui diz respeito à representação dos dados do registro usado em uma instância de domínio. A próxima seção trata de complementar esta descrição com a tradução LAL-DOM. Na seção 6.4. serão abordados aspectos da arquitetura referente aos níveis de domínio (generalização de instâncias de domínio) e aplicação (sistema de informação de uma instância de domínio), na exploração de perspectivas de reusabilidade.

6.3. Tradução LAL-DOM

A fim de que as informações do LAL possam ser representadas no DOM é importante estabelecer correspondências entre os elementos destes dois esquemas. A figura abaixo sumariza estas correspondências:

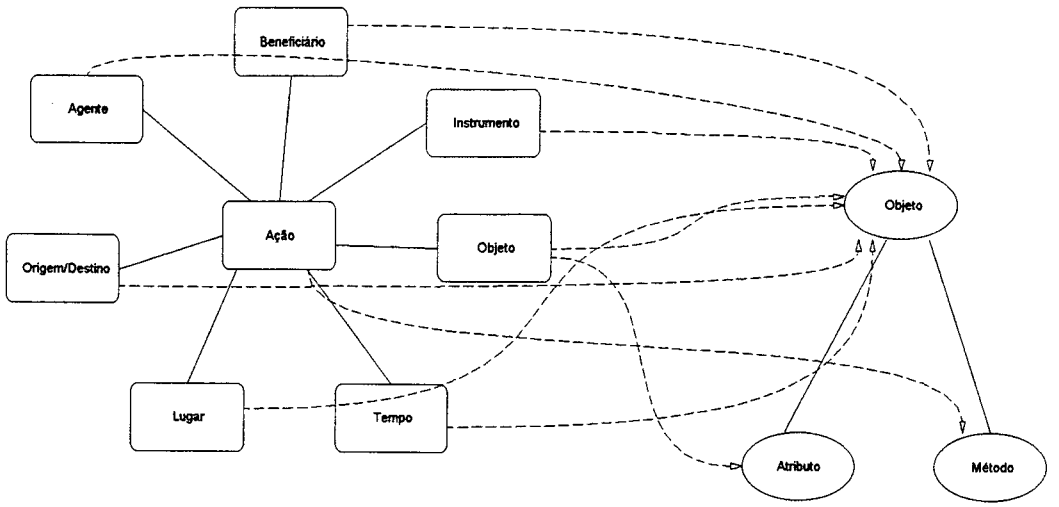

Figura 6.2.a Correspondências dos termos do LAL para os elementos do DOM: objeto, atributos e métodos

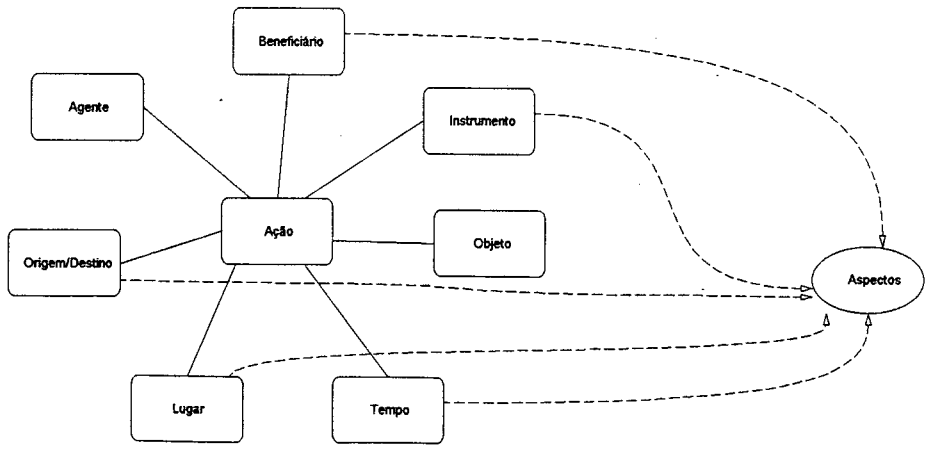

Figura 6.2.b. Correspondências dos termos do LAL para os elementos do DOM: aspectos

Para operacionalizar esta tradução, as correspondências foram traduzidas em regras. Estas regras foram aplicadas diretamente ao LAL, buscando, para cada termo do LAL, identificar os termos com os quais ele se relaciona e que tipo de relação existe. Além dessas correspondências, nos casos apropriados, buscou-se identificar relacionamentos de abstração. Abaixo um exemplo de um termo no LAL:

```
PROFESSOR (sujeito)

Noções:   -funcionário do CCAA que DÁ-AULA.

Impactos: -PROFESSOR PREPARA-SALA-DE-AULA.
          -PROFESSOR DÁ-AULA em SALA-DE-AULA, na HORA-
           DA-AULA, do LIVRO-N, do IDIOMA-M.
```

E a sua versão no DOM, depois de aplicadas as regras a este termo e a termos associados a ele:

```
PROFESSOR
DAR-AULA
   Lugar  (SALA-DE-AULA)
   Beneficiario  (ALUNO)
   Tempo  (HORA-DA-AULA)
   Instrumento (MATERIAL-DE-APOIO ATTENDANCE-CARD
                  SCHEDULE  VISUAL-AIDS )
   preCondicoes  (PREPARAR-SALA-DE-AULA)
PREPARAR-SALA-DE-AULA
   posCondicoes  (DAR-AULA)
   Lugar  (SALA-DE-AULA)
   Tempo  (antes da HORA-DA-AULA)
   Instrumento  (MATERIAL-DE-APOIO)
```

A tradução com auxílio das regras é feita de forma semi-automática, sendo solicitada a intervenção do analista sempre que houver ambigüidade. Além disso é necessário verificar se dados importantes não deixaram de ser traduzidos e adaptar algumas traduções. O aspecto `Tempo` do método `PREPARAR-SALA-DE-AULA` , por exemplo, foi adaptado para `(antes da HORA-DA-AULA)`, apesar da tradução pelas regras ter resultado em `(HORA-DA-AULA)`. Esta adaptação foi feita para manter o DOM mais fiel ao LAL.

De posse de uma representação dos dados sobre o registro, agora mais formal que o LAL, podemos trabalhar um aspecto importante da economia da comunicação: a reusabilidade.

6.4. Reusabilidade com o DOM

Na breve discussão feita anteriormente sobre esquemas semióticos, foi destacada sua característica social e sua interdependência ao uso de signos. Sendo o registro do usuário sobre o trabalho um esquema semiótico, ele herdará estas propriedades. Nós nos comunicamos porque, dentre outros fatores, compartilhamos esquemas semióticos e eles são em sua vasta maioria invariantes de um uso para outro.

Essa reusabilidade dos esquemas semióticos faz parte da economia da comunicação. Por outro lado, a Engenharia de Software tem como uma de suas principais questões a reusabilidade. Em relação ao desenvolvimento de software este aspecto tem grande importância, sendo-lhe atribuído desde o aumento de produtividade até a obtenção de um produto de maior qualidade.

As pesquisas em Reusabilidade (Arango & Prieto-Diaz, 1989) vêm buscando identificar que tipo de informação reusar e como reusá-la. Fontes identificadas para reuso vão desde código fonte e pessoas até módulos de software e linguagens de domínio. Nesta seção serão discutidas algumas formas de reuso que se pretende explorar com o DOM.

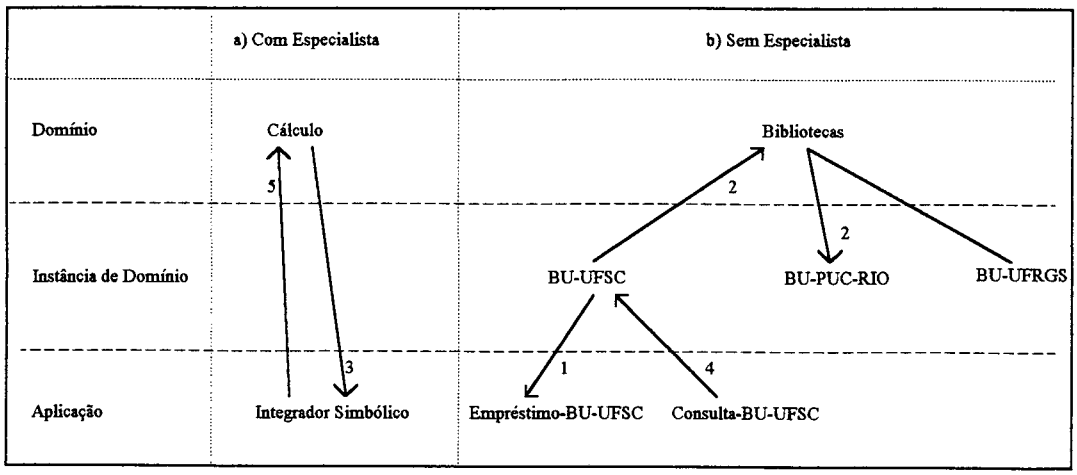

Figura 6.3. Reuso com o DOM

Os números na figura acima (Figura 6.3.) indicam possibilidades de reuso. O tipo 1 é o mais simples, ocorrendo no desenvolvimento de aplicações dentro de uma instância de domínio. A reusabilidade de informações entre instâncias de domínio (levando à construção de um modelo para o domínio) é indicada em 2. O tipo 3 é similar ao 1, só que ocorre em domínios já estáveis, geralmente com especialistas disponíveis e com literatura abundante. O desenvolvimento de uma aplicação específica pode enriquecer o registro, já que se dará em um contexto mais específico. Este tipo de reuso é apontado por 4 (para a instância de domínio) e 5 (para o domínio, no caso de domínios estáveis).

A reusabilidade do tipo 1 (em uma instância de domínio, para o desenvolvimento de aplicações) foi a explorada nos estudos de caso relatados no próximo capítulo. Este tipo de reuso tem como hipótese de trabalho que o mesmo registro usado para descrever a instância de domínio como um todo (Biblioteca Universitária da UFSC, por exemplo), pode ser usado como base para a Análise de Requisitos (ver seção 4.3. Análise de Requisitos e Interface) de uma aplicação específica (o Sistema de Empréstimo da BU, por exemplo). Alguns resultados preliminares com este tipo de uso serão relatados no próximo capítulo.

O reuso de informações entre instâncias de domínio (2) é um fator primordial na viabilização da estratégia. Caso este tipo de reuso não seja possível a cada nova instância de domínio todo o trabalho de elicitação terá que ser repetido. Aqui se trabalha com uma hipótese de trabalho de que grupos profissionais (bibliotecários, donos de banca de revistas, etc.) compartilham um mesmo registro lingüístico, ou linguagem profissional. Pretende-se com a elicitação do registro de várias instâncias em um mesmo domínio construir uma representação genérica estável ao domínio, a partir dos registros das instâncias de domínio. Até o presente momento o único domínio com mais de uma

instância trabalhada foi o das Bancas de Revista com duas instâncias (ver próximo capítulo).

Os tipos 4 e 5 de reuso dizem respeito à integração de um registro mais específico, que será usado na análise de requisitos de uma aplicação na instância de domínio (ou domínio) ao registro genérico elicitado e representado no LAL. Por exemplo, durante a análise de requisitos da aplicação Empréstimo da BU-UFSC, certos aspectos até então não observados ganharão relevância e um vocabulário específico será usado para se referir a estes aspectos. Possivelmente este terá elementos em comum com o registro elicitado a nível de instância de domínio, porém com novos elementos e refinamento de elementos já existentes. Este tipo de reuso ainda não foi explorado no contexto deste trabalho, mas se constitui no objeto de pesquisa de várias abordagens de Análise de Domínio (Prieto-Diaz, 1987).

Os mecanismos necessários para implementar estes tipos de reuso (com exceção do tipo 1) na forma de um sistema computacional dizem respeito a técnicas de Inteligência Artificial e fogem ao escopo deste trabalho.

Estes tipos de reuso são o principal objetivo do DOM e um dos fatores que poderão determinar o sucesso de uma estratégia deste tipo. Na seqüência, algumas discussões sobre os tópicos apresentados neste capítulo.

6.5. Discussão

O modelo DOM se diferencia de outras propostas de modelo para Análise de Domínio por não se preocupar em representar apenas elementos 'computáveis', mas os elementos pertinentes à descrição do registro. Este modelo se caracteriza, assim, como contribuindo para uma abordagem de análise de domínio voltada não apenas para a reusabilidade na construção de programas, mas com a reusabilidade associada à comunicação.

Um dos usos possíveis do DOM é por analistas que não sejam aqueles que elicitaram o LAL. Como este modelo é menos ambíguo, deve possibilitar o entendimento por aqueles que não compartilham o mesmo contexto e *background* daqueles que elicitaram um dado registro.

A arquitetura do modelo é um elemento aberto e ainda não estável. As diversas experiências que vêm sendo realizadas com o modelo têm sugerido alterações que vão sendo incorporadas. Um exemplo disto é o uso da teoria de Gramática de Casos para categorizar os termos do LAL, que gerou a necessidade de se estabelecer um mapeamento destes no DOM.

Os objetivos de reusabilidade ainda foram pouco explorados, embora alguns resultados preliminares sejam relatados no próximo capítulo, juntamente com o objetivo de validação do LAL através da tradução. Os estudos de caso também trabalharam com a

validação da construção de um *framework* de classes para desenvolvimento de aplicações e demonstraram que o registro é uma boa fonte de critérios para determinar que elementos são importantes do ponto de vista do usuário.

O próximo capítulo vai tratar de fornecer uma visão unificadora da estratégia de software centrada no registro, através do relato dos estudos de caso e discussão de resultados.

7. Uma Estratégia de Software Centrada no Registro: Estudos de Caso

Nos dois capítulos precedentes foram definidos dois espaços para expressão de informação, LAL e DOM. Para torná-los usáveis, é preciso descrever um processo que estabeleça uma maneira de criar e transformar estes espaços de informação, e adaptá-los aos métodos convencionais de desenvolvimento de software.

O objetivo deste capítulo é descrever uma estratégia de software baseada no registro, que através dos instrumentos já descritos, consiga fornecer suporte à comunicação no fazer e usar software. Esta estratégia será ilustrada por exemplos retirados dos dois estudos de caso executados por alunas do último ano do curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Primeiramente, o contexto dos estudos de caso é estabelecido e, depois, aspectos da estratégia abordados. A estratégia começa com a elicitação do LAL, em entrevistas com um ator da instância do domínio; continua com a tradução LAL-DOM, através de regras e com a colaboração do analista; e finalmente desemboca no uso de um método desenvolvimento de software, onde os requisitos serão expressos em termos do DOM.

7.1. Contexto dos Estudos de Caso

Nos últimos anos, desde a proposta de Linguagens da Aplicação (Leite, 1989), alguns estudos têm sido realizados na elicitação destas linguagens. Destes estudos alguns foram executados por alunos do último ano do curso de Ciências da Computação da UFSC. Nemes & Tagliari (1992) foi o primeiro destes trabalhos e serviu como uma introdução ao assunto. O trabalho de Mosqueta & Colagrande (1993) foi co-orientado pelo autor e serviu como estudo-piloto na elicitação do LAL e em uma primeira tentativa de tradução para o modelo DOM.

Da experiência obtida com estes trabalhos, dois estudos de caso foram propostos e orientados pelo autor. Estes trabalhos, relatados em Pereira & Moro (1994) e Treter & Petry (1994), foram mais longe na proposta de se usar conhecimento do registro para construção de programas, chegando à construção de um protótipo para uma aplicação.

O estudo-piloto citado acima se realizou em uma instância do domínio das Bancas de Revista, denominada Banca de Revistas do Marcelo. Para que a reusabilidade entre instâncias de domínio possa ser explorada no futuro (seção 6.4. Reusabilidade com o DOM), um novo estudo em bancas de revista foi proposto e executado (Pereira & Moro, 1994), na instância de domínio Banca da Praça XV, de agora em diante denominada Banca.

Já Treter & Petry (1994) realizaram um estudo em uma instância de um domínio aparentemente mais complexo que uma banca de revistas, que é o das Escolas de Idiomas. A instância de domínio trabalhada foi o CCAA-Centro, que será referenciada como CCAA.

Em seguida será descrita estratégia centrada no registro, e a execução das várias etapas dos estudos de caso e aspectos ilustrativos serão discutidos

7.2. Estratégia Centrada no Registro e Estudos de Caso

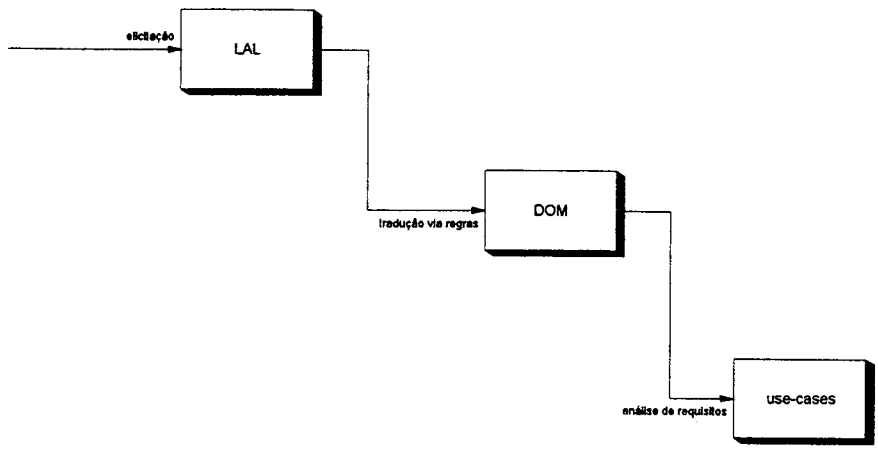

Figura 7.1. Diagrama da Estratégia Centrada no Registro

Os estudos de caso foram executados seguindo o diagrama acima. As principais atividades foram a elicitação do LAL, a tradução para o modelo DOM e a análise de requisitos usando a técnica de *use cases* (Jacobson, 1993). A partir dos *use cases*, protótipos foram construídos para investigar se a descrição das necessidades do usuário obtida é compatível com a construção de programas.

Os itens a seguir descrevem aspectos relevantes de cada uma destas etapas para os estudos de caso da Banca e do CCAA.

7.2.1. Elicitação do LAL

Os analistas de sistemas costumam ver suas atividades como construção de modelos de uma realidade objetiva e voltados para a resolução de algum problema. As alunas envolvidas nos estudos de caso não foram exceção e, para que a elicitação do LAL ocorresse de forma apropriada, esta visão associada à atividade do analista de sistemas teve que ser quebrada: a realidade tida como objetiva deve ser substituída pela realidade do ponto de vista do usuário; o conhecimento sendo levantado não objetiva resolver um problema específico, mas familiarizar o analista com o registro do usuário.

Para salientar esta distinção, elas desenvolveram uma análise de sistemas típica antes da elicitação. Esta serviu como um primeiro contato e como forma de contrastar a visão das analistas com a do usuário.

Em seguida as analistas receberam material bibliográfico sobre o LAL e, de acordo com a demanda, algum material complementar. Este material foi lido e discutido e serviu de suporte às primeiras entrevistas com o usuário escolhido como informante, das quais resultou uma primeira versão do LAL para cada um dos ambientes. Este LAL foi construído de acordo com a proposta original de (Leite, 1989) (ver seção 5.3.).

A situação típica das primeiras tentativas de construção do LAL é de impasse, onde literalmente 'não se sabe por onde começar'. O grupo que elaborou o estudo da Banca veio com uma solução inovadora: não começar diretamente do LAL, mas da construção de frases que então são analisadas para identificação dos termos do LAL. Abaixo alguns exemplos de frases (Pereira & Moro, 1994, p. 14):

> O fornecedor faz o cadastro da Banca.
>
> O proprietário fornece as informações referentes à Banca para o preenchimento do cadastro no fornecedor.
>
> Todos os dias úteis, o proprietário pega o reparte, por consignação, nos fornecedores de revistas e livros e recebe a nota de entrega e a nota de devolução do fornecedores de revistas e livros.

Estas frases foram então analisadas e delas extraídas os termos que formaram uma versão inicial do LAL/Banca, por exemplo (Pereira & Moro, 1994, p. 14):

```
PEGAR-REPARTE (AÇÃO)
Noções:
        1. Ato realizado pelo PROPRIETÁRIO no FORNECEDOR-DE-
        REVISTAS-LIVROS
        2. Ocorre todos os dias úteis
        3. Ato realizado por CONSIGNAÇÃO
Impactos:
        1. REVISTAS e LIVROS ficam DISPONÍVEIS-PARA-VENDA
        2. O PROPRIETÁRIO recebe a NOTA-DE-ENTREGA e a NOTA-DE-
        DEVOLUÇÃO do FORNECEDOR-DE-REVISTAS-E-LIVROS
```

Esta abordagem resultou em conjunto inicial de termos que foi depois validado e incrementado. Ela merece destaque por se tratar de uma maneira de quebrar o impasse inicial, que geralmente ocorre com iniciantes na técnica.

A elicitação do LAL se deu concomitante com o entendimento mais preciso dos conceitos envolvidos, o que gerou diversas versões do LAL para cada um deles. Para cada versão construída e validada informalmente com os usuários foi gerada uma nova versão do LAL, até a versão considerada como final, quando foi alcançado consenso entre analistas e usuários.

Foi durante este processo que foram efetuadas as alterações no esquema de classificação dos termos do LAL, já apresentado na seção 5.3. Estas alterações se refletiram também na ferramenta HyperLex (Leite & Franco, 1990), que foi reimplementada dentro do contexto deste trabalho.

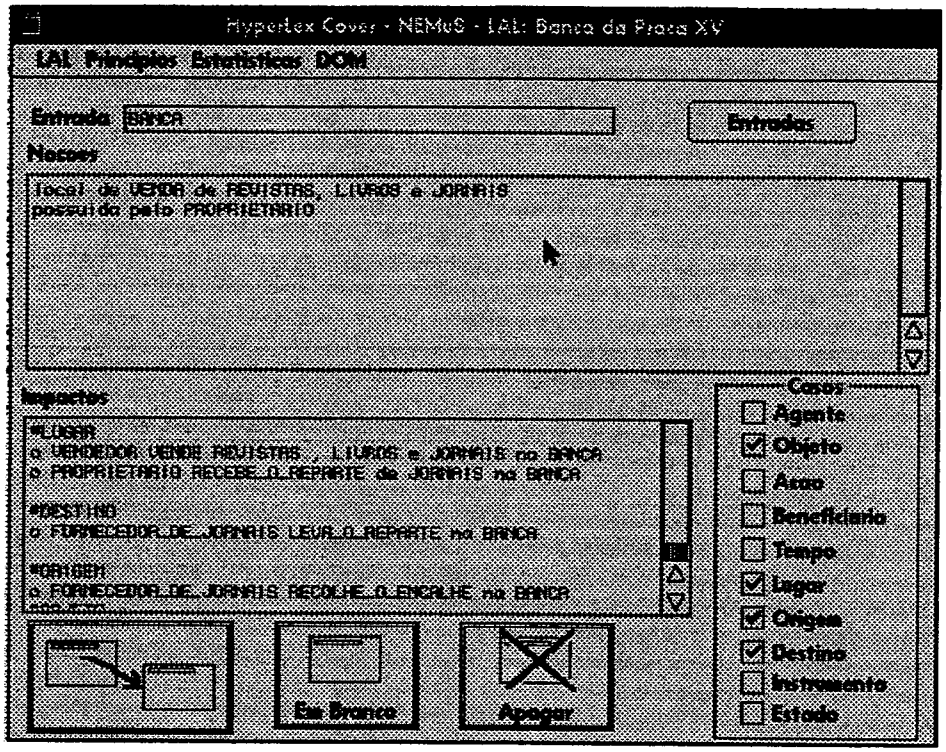

Figura 7.2. HyperLex com casos

Esta ferramenta (Figura 7.2.) tem por objetivo possibilitar a autoria navegação pelo LAL como um hipertexto. Ela pode também verificar algumas propriedades desejáveis do LAL, como por exemplo a circularidade.

A versão do LAL considerada como final pelos dois grupos está de acordo com a proposta modificada para incorporar casos (ver seção 5.3.). A partir desta versão foi feito o mapeamento para o DOM, como descrito a seguir.

7.2.2. Tradução LAL-DOM

À versão validada do LAL/Banca e LAL/CCAA foram aplicadas regras de tradução. Estas regras seguiram as correspondências estabelecidas entre termos do LAL e elementos do DOM, com informações adicionais fornecidas pelas analistas. A aplicação das regras resultou na criação de dois modelos: DOM/Banca e DOM/CCAA. Uma porção do DOM/CCAA pode ser visualizada na figura abaixo:

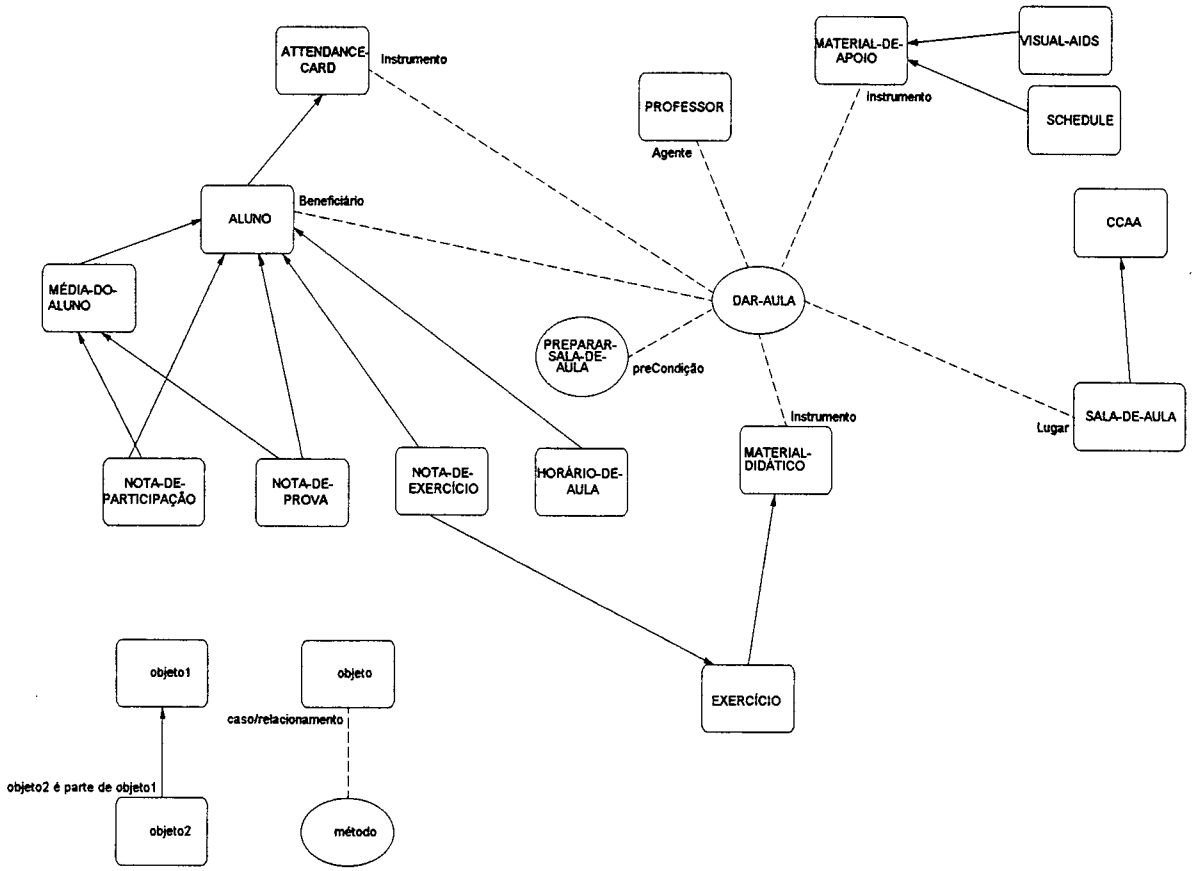

Figura 7.3. Visão parcial do DOM/CCAA

Na Figura 7.3. dois métodos foram explicitados para exemplificar o mapeamento LAL-DOM. O foco principal é o método DAR-AULA, correspondente à ação de mesmo nome no LAL. Lê-se: 'o PROFESSOR executa a ação DAR-AULA. Antes disso, a ação PREPARAR-SALA-DE-AULA deve ter sido executada. Para DAR-AULA são usados o MATERIAL-DE-APOIO, o MATERIAL-DIDÁTICO e o ATTENDANCE-CARD. A ação de DAR-AULA é executada na SALA-DE-AULA e seu beneficiário é o ALUNO". Vejamos agora a entrada DAR-AULA do LAL/CCAA:

DAR-AULA\MINISTRAR-AULA\DÁ-AULA (ação)

Noções:

1. ação executada pelo PROFESSOR.
2. é realizada quando ALUNO está em SALA-DE-AULA.
3. é realizada na HORA-DA-AULA.
4. PROFESSOR usa ATTENDANCE-CARD.
5. PROFESSOR utiliza MATERIAL-DE-APOIO.

Impactos:

1. PROFESSOR PREPARA-SALA-DE-AULA.

Podemos observar que as noções são mapeadas de forma bastante fiel. Já o impacto é bastante ambíguo: a sala de aula deve ser preparada antes, depois ou durante a ação de dar aula ? É neste tipo de situação que a intervenção do analista é solicitada, escolhendo uma dentre várias 'interpretações' do LAL, neste caso a de que PREPARAR-SALA-DE-AULA deve ocorrer antes de DAR-AULA (pré-condição).

A tradução para o DOM ocasionou, nos dois estudos de caso, uma nova versão do LAL. Isto está de acordo com um dos objetivos do DOM, que é de eliminar falhas na elicitação do LAL, fazendo com que as definições sejam claras. Um exemplo de omissão ocorreu com o termo MATERIAL-DE-APOIO do LAL/CCAA, que aparecia como um instrumento sem nenhuma ação associada. Este problema foi apontado durante a tradução e as correções necessárias foram efetuadas.

Os outros dois objetivos, de reusabilidade do registro e de prover um framework de classes para a construção de aplicações, estão relacionados com o item a seguir, onde é descrita a construção de aplicações tendo como base o conhecimento do registro do usuário sobre o trabalho.

7.2.3. Prototipação de uma Aplicação: DOM e *use cases*

Dos vários tipos de reuso das informações discutidos na seção 6.4. Reusabilidade com o DOM, o tipo 1 é o que está relacionado com a construção de aplicações. Este tipo de reuso prevê que o conhecimento existente sobre o registro do usuário sobre o trabalho na instância de domínio seja (re)usado na construção de aplicações.

Foi notado em Pereira & Moro (1994) que o uso deste conhecimento ocorre antes mesmo da construção da aplicação, na escolha de que aplicação será desenvolvida. Esta decisão é discutida em termos do registro, e resultou para a Banca em desenvolver um Controle de Vendas de Jornais e para o CCAA no Sistema de Matrícula.

O objetivo da estratégia centrada no registro é prover suporte à **conversação** que se dá entre analista e usuário para inventar signos baseados em computador. A partir do momento em que estes signos são inventados é preciso construir os programas cujos processos são a substância da expressão, que na interpretação pelo usuário podem se transformar em forma da expressão e ter seu conteúdo estabelecido (Andersen, 1990). Neste momento da construção de programas, a ênfase no aspecto de processo em relação ao software pode ser minimizada e a antiga ênfase no aspecto produto retomada, onde qualidades tais como extensibilidade, robustez e legibilidade, mais relacionadas com a atividade de quem constrói programas, podem ser maximizadas.

Esta ênfase no aspecto produto pode ser encontrada na vasta maioria dos métodos de desenvolvimento de software (Floyd, 1988). Para os estudos de caso o método escolhido foi Objectory (Jacobson e outros, 1993). Este método procura resolver problemas de outros métodos de desenvolvimento orientados a objetos através da noção de *use case*, a qual dirige todo o desenvolvimento.

Associado ao conceito de *use case* temos o de ator. Ator representa algo que interage com o sistema. Um *use case* por sua vez é um curso completo de eventos iniciados por um ator e especifica a interação que tem lugar entre um ator e o sistema (Jacobson e outros, 1993).

Os *use cases* foram projetados para que todo o desenvolvimento do software seja embasado neles: análise (construção de *use cases*), projeto (*use cases* em uma determinada plataforma de software e hardware), implementação (código fonte que implementa os *use cases*), testes (verificar se os *use cases* de uma aplicação funcionam separadamente e em conjunto) e manutenção ou evolução (alterar o *use case* e propagar esta alteração para os outros modelos).

Para cada aplicação trabalhada nos estudos de caso um conjunto de *use cases* correspondente foi definido. Abaixo, exemplo com os *use cases* para o Sistema de Matrícula do CCAA:

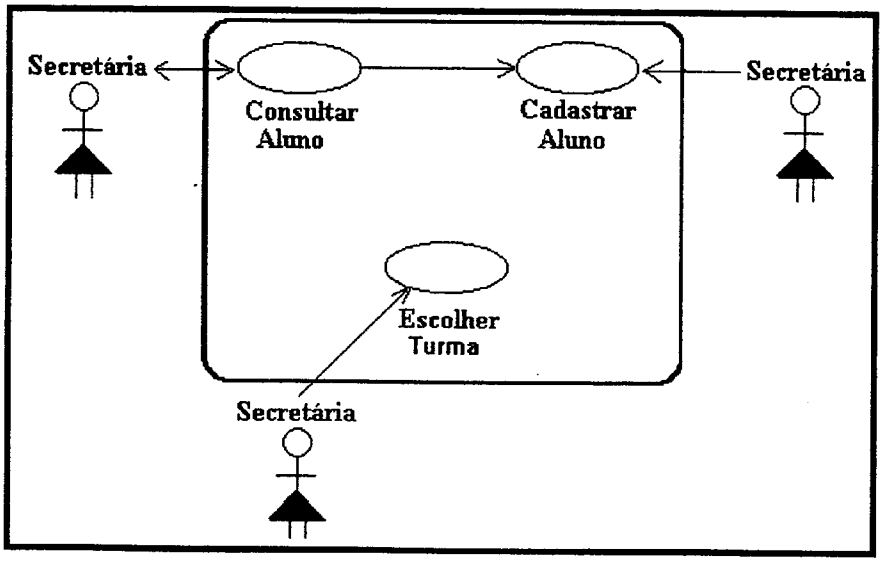

Figura 7.4. Atores e use cases da aplicação Sistema de Matrícula - CCAA.(Treter & Petry, 1994, p. 25)

Na Figura 7.4. podemos observar o ator que interagirá com o Sistema de Matrícula, a secretária, e três *use cases*: Consultar Aluno, Cadastrar Aluno e Escolher Turma. Cada um destes *use cases* possui uma descrição em linguagem natural. Abaixo, exemplo do use case Cadastrar Aluno:

- **Use case Cadastrar Aluno (Figuras 7.5.a e 7.5.b)**

-Secretária aperta o botão CADASTRAR.
-Sistema atribui um número ao aluno (número de matrícula).
-Secretária pergunta o nome ao aluno, e digita.
-Se aluno já cadastrado, então sistema mostra mensagem "Aluno já está cadastrado".
-Se aluno não cadastrado, Secretária pergunta e digita dados do aluno (NOME COMPLETO, RUA, APTO, BAIRRO, CIDADE, ESTADO, CEP, TELEFONE, RG, GRAU DE ESCOLARIDADE [1° GRAU COMPLETO, 1° GRAU INCOMPLETO, 2° GRAU COMPLETO, 2° GRAU INCOMPLETO, 3° GRAU COMPLETO, 3° GRAU INCOMPLETO], ESCOLA, PROFISSÃO, LOCAL DE TRABALHO, LOCAL DE NASCIMENTO, DATA DE NASCIMENTO, FILIAÇÃO).
-Secretária conclui o cadastro (Botão OK) ou cancela (Botão CANCELA).

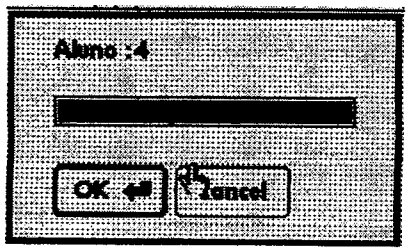

Figura 7.5.a. Use case Cadastrar Aluno.

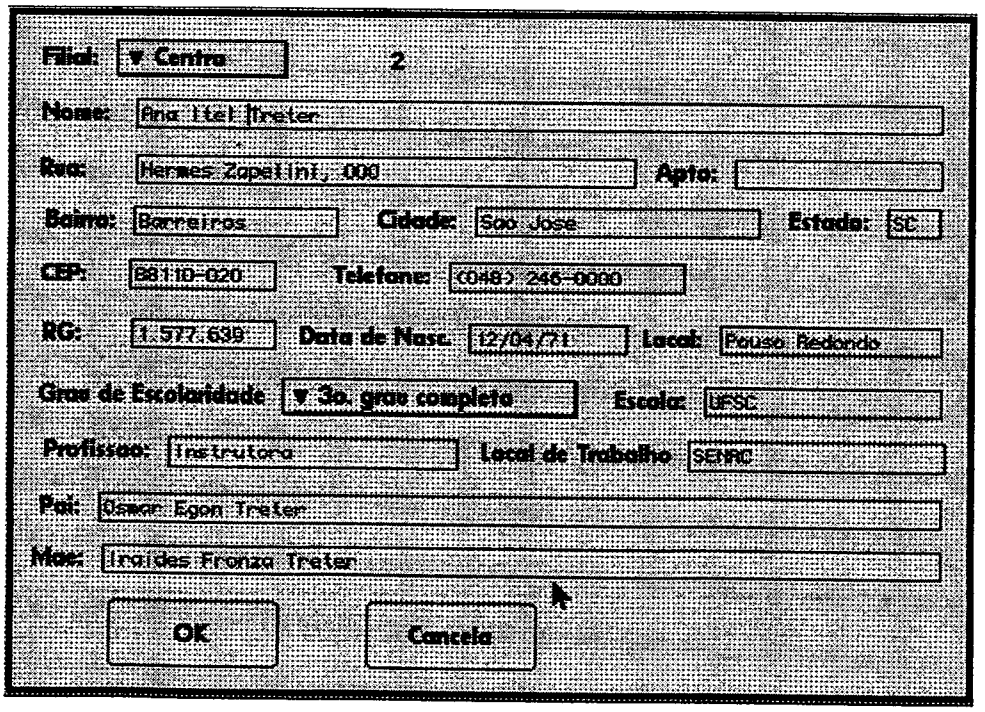

Figura 7.5.b. Use case Cadastrar Aluno (Treter & Petry, 1994, p. 26)

Depois de especificados os *use cases* e suas interfaces, um conjunto de objetos é estabelecido para suportá-los e implementado em uma linguagem de

programação orientada a objetos, neste caso o Smalltalk/V-286. Deste conjunto de objetos, os do tipo entidade vieram todos do registro elicitado e representado no DOM, sendo que a identificação destes objetos é um problema importante em Análise Orientada a Objetos.

7.3. Discussão

Do ponto de vista da Semiótica Computacional (Andersen, 1990), os *use cases* podem ser vistos como signos compostos baseados em computador. Estes signos compostos foram inventados em uma conversação entre analista e usuário usando como sistema de significação o registro sobre o trabalho. Para um dos grupos (CCAA) foram feitos registros das conversações com o usuário para definição dos *use cases* e pode ser observada intimidade das analistas em tratar com os termos do registro do usuário.

Já que estes signos foram discutidos e propostos em conjunto pelo usuário e analista, de uma forma que se acredita compreensível para o usuário, o aspecto de interpretação destes signos deve ser facilitado.

Outra diretriz foi usar o máximo possível termos do registro para nomear e descrever os *use cases*. Desta resultou que pelo menos um dos termos do nome de cada use case pertence ao registro do usuário. Este resultado parece de acordo com a noção de *gaps* no registro (Andersen, 1990), onde novos termos são criados para uma nova forma do conteúdo. Como o objetivo dos use cases é descrever uma situação futura, é natural que novos termos surjam para designar estes novos processos. Usar a noção de *gaps* melhora a probabilidade destes serem incorporados ao registro do usuário.

Apesar de o registro usado ter sido o sobre o trabalho, os nomes dos *use cases* foram intuitivamente usados para rotular botões na interface. Estes serão signos candidatos ao registro baseado no computador, que envolve o registro de trabalho e os signos da interface. É preciso um processo de prototipação, no entanto, para verificar a válida interpretação e a interferência destes no registro baseado no computador, especialmente por sua origem no registro sobre o trabalho.

Em relação ao LAL e DOM, é preciso recapitular que seu atual estágio de desenvolvimento se deveu aos estudos de caso, que trouxeram à tona problemas e restrições destes instrumentos.

8. Discussões

Este trabalho abordou o fenômeno software de uma perspectiva semiótica e com a preocupação de estabelecer maneiras de compreendê-lo e tratá-lo através da relação com um esquema semiótico específico: o registro do usuário sobre o trabalho.

Para tal, os objetivos foram estabelecidos no capítulo 1. Introdução, e permearam todo o trabalho. As seções seguintes tratarão de discutir os resultados em relação aos objetivos, trabalhos relacionados e possíveis continuações nesta linha de pesquisa.

8.1. Dos Objetivos

O objetivo geral, de *explorar uma estratégia de software centrada no registro através de estudos de caso* foi decomposto em objetivos específicos:

- *Estabelecer relações entre registro(s) e software, na construção e uso de programas de computador.*

Estas relações foram discutidas no capítulo 4. Software como Construção de Significado. Foram identificadas na literatura dois tipos de registro que interagem com o software: o registro de trabalho quando do uso e o registro sobre o trabalho quando da concepção. O registro sobre o trabalho foi escolhido como o foco desta pesquisa, já que o enfoque é a comunicação quando da concepção do software (buscando uma melhor comunicação quando do uso, pois a primeira estabelece o funcionamento da segunda).

- *Incrementar e desenvolver técnicas para levantamento do conhecimento sobre o registro lingüístico.*

Tendo estabelecido o conhecimento do registro sobre o trabalho como básico à concepção de software, técnicas para levantar este conhecimento são necessárias. Estas foram tratadas especialmente no capítulo 5. Léxico Ampliado da Linguagem, onde as técnicas originalmente propostas pelo Prof. Júlio Leite foram expostas e expandidas à luz das experiências nos estudos de caso.

- *Investigar formas de (re)uso do conhecimento sobre o registro na construção de programas.*

Através da construção de um protótipo de aplicação para cada instância de domínio estudada, procurou-se demonstrar como o conhecimento sobre o registro pode ser usado na construção de programas de forma a melhorar as probabilidades de os signos propostos serem adequadamente interpretados, já que o usuário os inventa junto com o analista. As outras formas de reuso proposta na seção 6.4., ainda não foram exploradas e devem ser objeto de pesquisas adicionais.

8.2. Trabalhos Relacionados

Nos itens a seguir serão discutidos alguns trabalhos relacionados com a presente abordagem. Os quatro paradigmas para desenvolvimento de sistemas de informação (Hirschheim & Klein, 1989) são discutidos no item 8.2.1. No item 8.2.2. a noção de *breakdowns* como base para o design (Winograd & Flores, 1985) é explorada em relação à elicitação do LAL e Análise de Requisitos.

8.2.1. Quatro Paradigmas

Hirschheim & Klein (1989) em seu artigo entitulado *Four Paradigms of Information Systems Development* propõem quatro paradigmas para classificar os métodos de desenvolvimento de sistemas de informação. Esta classificação originou-se da constatação de que embora exista uma corrente ortodoxa de trabalhos nesta área, existem trabalhos alternativos recentes que fogem a esta corrente, se baseando em um conjunto fundamentalmente diferente de pressupostos. Cada método traz consigo um conjunto de pressupostos implícitos ou explícitos "sobre a natureza das organizações humanas, a natureza da tarefa de design e o que se espera deles (projetistas)" (Hirschheim & Klein, 1989, p. 1199). Estes pressupostos, segundo eles, têm as seguintes características:

- lidam com atitudes em relação à realidade e como obter conhecimento sobre ela;
- ao se adotar uma abordagem específica se estão adotando estes pressupostos implícita ou explicitamente;
- as maneiras como os objetivos do sistema são legitimados estão diretamente ligadas à abordagem adotada;
- importantes conseqüências sociais resultam da adoção de uma abordagem.

Já que os pressupostos se relacionam com a realidade e como obter conhecimento, eles foram organizados em dois eixos: o ontológico e o epistemológico. Nos extremos do eixo ontológico estão o objetivismo (aplicar métodos das ciências naturais para abordar fenômenos humanos) e o subjetivismo (nega o uso de métodos das ciências naturais e busca um entendimento baseado na experiência subjetiva dos indivíduos). No eixo epistemológico temos ordem (vê o mundo social como ordenado, estável, consensual, integração e coordenação funcional) e conflito (mudança, conflito, desintegração e coerção). Destes eixos resulta um plano onde os quatro paradigmas são estabelecidos:

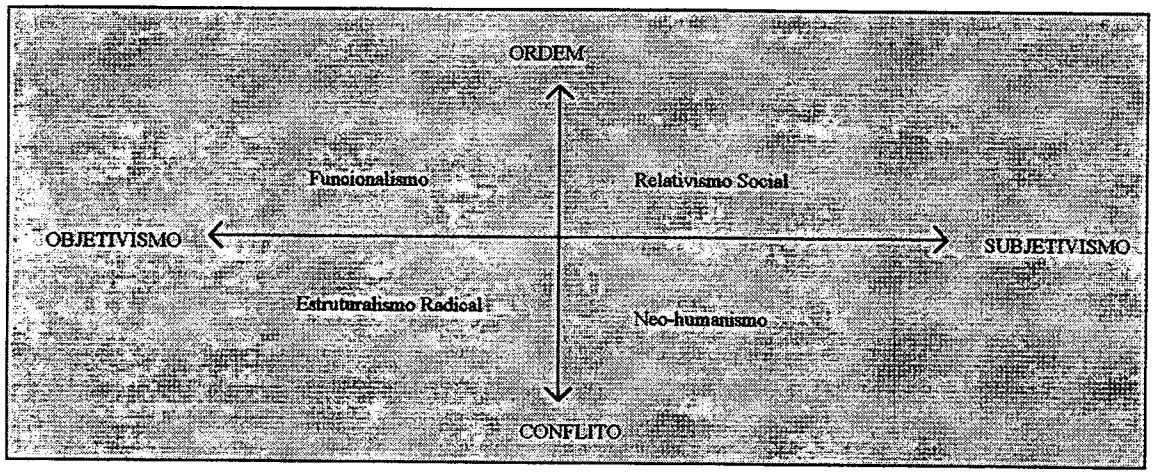

Figura 8.1. Paradigmas de Desenvolvimento de Sistemas de Informação (Hirschheim & Klein, 1989, p. 1202)

Os pressupostos associados a cada um dos paradigmas são:

- *Funcionalismo*: epistemologia do positivismo (desenvolvedor busca ganhar conhecimento sobre a organização através de um conjunto de relações de causa-efeito) e ontologia do realismo (assume a existência de uma realidade externa independente do observador).

- *Relativismo Social*: epistemologia do anti-positivismo (nega a busca de explicações causais e empíricas para fenômenos sociais) e ontologia do nominalismo (não há uma realidade 'lá fora', mas sim uma realidade socialmente construída). A realidade social é explicada do ponto de vista dos agentes organizacionais envolvidos na construção da realidade.

- *Estruturalismo Radical*: epistemologia do positivismo (na forma específica da visão materialista da história e da sociedade) e ontologia do realismo. Procura criticar o *status quo* com o objetivo de prover *rationale* para uma mudança radical.

- *Neo-humanismo*: epistemologia de dois tipos, positivismo (para controle técnico) e anti-positivismo (para conhecimento relacionado ao entendimento mútuo e emancipação. Ontologia também de dois tipos, realismo (para aspectos técnicos) e nominalismo (para entendimento mútuo e emancipação). O desenvolvedor busca libertar os atores da organização das limitações de problemas na comunicação e criação de sistemas que suportem um discurso racional, que permita emancipação dos indivíduos e crescimento pessoal. Isto só pode ser obtido através de imersão do desenvolvedor na organização.

O presente trabalho descarta dois paradigmas, o do funcionalismo (onde a maioria dos métodos tradicionais se encaixa) e o estruturalismo radical, por assumir pressupostos epistemológicos mais próximos do subjetivismo. A orientação do desenvolvimento pelo registro do usuário (ou sua visão da realidade) confirma isso.

O neo-humanismo clama resultados que, de acordo com os próprios autores, ainda estamos longe de alcançar. Resta o relativismo social, onde a presente abordagem se encaixa mais facilmente.

Os autores do trabalho citado dão o subtítulo de *Systems Development as Sense Making* a este paradigma e denominam o desenvolvedor de facilitador. Esta é uma denominação apropriada em nossa abordagem onde o usuário participa da invenção dos signos baseados em computador e o analista atua como um facilitador entre o usuário e a tecnologia. O objetivo do desenvolvimento é alcançar consenso ou aceitação do sistema (programa interpretado como o software desejado). A maneira apontada para se alcançar este consenso é através de interação entre todas partes envolvidas (interação envolve comunicação e comunicação pressupõe um sistema de significação, que em nosso trabalho foi o registro do usuário sobre o trabalho).

8.2.3. *Breakdowns* e Design

Winograd & Flores (1985) em seu livro *Understanding Computers and Cognition: A New Foundation for Design*, apresentam o conceito de *breakdown*, vindo do trabalho filosófico de Heidegger, como uma das bases de sua proposta de design.

O exemplo dado é de uma pessoa pregando algo. Enquanto tudo corre bem o martelo está *ready-at-hand*, significando que a pessoa não tem conhecimento dele mais do que tem dos tendões em seu braço. Se algo dá errado, no entanto, como errar o prego ou a cabeça do martelo sair voando, o martelo passa a existir conscientemente para a pessoa pregando. Aconteceu um *breakdown*. Assim, os objetos e propriedades não existem inerentemente no mundo, mas emergem se tornando *present-at-hand* na ocorrência de um *breakdown*, que só é possível em uma atividade em que estejamos engajados e que tenha potencial para tal.

É somente através de *breakdowns* que tomamos conhecimento de objetos e nos referenciamos a eles através de linguagem e é neles que as atividades design podem ser baseadas. A elicitação do LAL junto aos usuários foi um breakdown provocado junto ao usuário: 'explique-me seu trabalho'. Através dele objetos do trabalho do usuário tiveram que ser explicitados e foram registrados no LAL. A conversação para a construção de uma aplicação específica dentro da instância de domínio tratada na elicitação do LAL, se dá em um contexto diferente de breakdown, fazendo emergir novos objetos. Assim, trabalha-se como um processo de design de software baseado em breakdowns graduais e da perspectiva do usuário. Quando um analista se propõe a analisar um sistema sozinho ele pode usar o seu domínio de breakdown para relevar os aspectos importantes de uma dada realidade e que não correspondem àqueles importantes para os usuários

8.3. Continuação

As duas principais linhas de trabalho que podem dar continuidade à proposta uma estratégia centrada no registro dizem respeito à reusabilidade e à relação do registro sobre o trabalho e registro de trabalho.

Na linha da reusabilidade diversas modalidades de trabalho podem ser desenvolvidas, das quais destacamos:

- dois grupos numa mesma instância de domínio, para verificar qual a influência dos analistas no resultado do processo de elicitação do LAL;

- um grupo faz levantamento do registro e outro a análise de requisitos, usando os resultados do primeiro;

- estudar um número maior de instâncias de domínio em um mesmo domínio, para verificar a reusabilidade entre instâncias de domínio;

Quanto à relação entre os registros sobre e de trabalho é importante estudar as diferenças e semelhanças entre estes registros e se os signos propostos no contexto do registro sobre o trabalho funcionam no contexto de uso do programa, onde o registro de trabalho é usado.

9. Referências Bibliográficas

ANDERSEN, P. B. A Theory of Computer Semiotics. Cambridge: Cambridge University Press, 1990, 416 p.

ANDERSEN, P. B. The force dynamics of interactive systems. Towards a computer semiotics. Department of Informatic and Media Science - University of Aarhus, 1992, 36 p.

ARANGO, G. and PRIETO-DÍAZ, R. Domain Analysis: Concepts and Research Directions. Computer Society Press Tutorial, Pittsburgh, 1989.

BARTHET, M.-F. Logiciels Interactifs et ergonomie: modèles e méthodes de conception (first ed.). Paris:Bordas, 1988.

BROWNE, D. P. The user interface: the poor relation in structured methods. in: Hartson, H.R. & Hix, D. (eds.). Advances in Human Computer Interaction. Norwood, New Jersey: Ablex Publishing Corporation, 1988, v. 3.

COUTAZ, J. Interfaces hommes-ordinateurs: conception et réalisation. (first ed.). Paris: Bordas, 1990.

CYBIS, W. de A. A Identificação dos Objetos de Interfaces Homem-Computador e seus Atributos Ergonômicos. Tese de Doutorado em Engenharia de Produção, UFSC, 1994.

DE SOUZA, C. S. The semiotic engineering of user interface languages. Int. J. Man-Machine Studies (1993) 39, 753-773

DE SOUZA, C. S. Testing Predictions of Semiotic Engineering in Human-Computer Interaction. Anais do VIII Simpósio Brasileiro de Engenharia de Software, Curitiba, 25-29 de outubro de 1994.

DUBOIS, J. e outros. Dicionário de Lingüística. São Paulo: Cultrix, 1991.

ECO, U. A Theory of Semiotics. Indiana: Indiana University Press, 1979, 354 p.

FAUST, R. Contribuições da Semiótica e Psicologia Cogntiva à Construção de Software: Uma Investigação. Projeto de conclusão de curso. UFSC- Florianópolis, 1992.

FILLMORE, Ch. J. The case for case. In: E. Bach & R.T. Harms (eds.), Universal in Linguistic Theory. 1-90. London: Holt, Rinehart and Winston, 1968.

FLOYD, C. Outline of a Paradigm Change in Software Engineering. Software Engineering Notes, ACM SIGSOFT, April 1988, pp. 25-37.

FOLEY, J.D. & VAN DAM, A. Fundamentals of interactive computer graphics. (first ed.) Readings, Massachusetts: Addison-Wesley Publishing Company, 1982.

FRANCO, A. P. & LEITE, J. Uma Estratégia de Suporte à Engenharia de Requisitos. XII Congresso da Sociedade Brasileira de Computação, Outubro 1922, pp. 200-213.

GREIMAS, A.J. & COURTÉS, J. Dicionário de Semiótica. São Paulo: Cultrix, 1979.

HIRSCHHEIM, R. & KLEIN, H. K. Four Paradigms of Information Systems Development. Communications of the ACM, v. 32, n. 10, pp. 1199-1216, 1989.

HJELMSLEV, L. Prolegômenos a uma Teoria da Linguagem. São Paulo, Perspectiva, 1975.

HOLMQVIST, B. & ANDERSEN, P.B. Work Language and Information Technology. Journal of Pragmatics, 11, pp. 327-357, 1987.

HYMES, D. The Ethnography of Speaking. In: Fishman, J. A. ed. Readings in the sociology of language. The Hague, Mouton, 1968, pp. 99-138.

JACOBSON, I. Object-Oriented Software Engineering - A use case driven approach. Reading, Massachusetts: Adisson-Wesley Publishing Company, 1993.

KAMERSGAARD, J. Four different perspectives on human-computer interaction. International Journal of Man-Machine Studies, **28**, 343-362.

KATZENBERG, B. & PIELA, P. Work Language Analysis and the Naming Problem. Communications of the ACM, v. 36 n. 4., pp.86-92, 1993.

KUHN. T.S. A Estrutura das Revoluções Científicas.

LAMSWEERDE, A. e outros. A Conceptual Meta-model for Representing Product-level Knowledge in Requirements Acquisition. Report no. 7. Institut d'Informatique, Facultés Univesitaires de Namur, 1989.

LEITE, J. & FRANCO, A. O Uso de Hipertexto na Elicitação de Linguagens de Aplicação. Anais do IV Simpósio Brasileiro de Engenharia de Software, SBC, Águas de São Pedro, SP, 1990, pp. 135-149.

LEITE, J. Elicitation of Application Languages. Monografias em Ciência da Computação, PUC-RIO, n. 30, 1989.

LEITE, J. Comunicação Pessoal. 1993.

LEMOS, A. Um estudo preliminar e uma experiência na integração entre a concepção formal de sistemas e a construção de programas. Dissertação de Mestrado. Porto Alegre: Instituto de Informática, UFRGS, 1991.

MATTOS, N. An Approach to Knowledge Base Management. University of Kaiserslautern, Germany, 1989.

MEYER, B. Object-Oriented Software Construction. Prentice-Hall, New Jersey, 1988.

MOSQUETA, A. P. & COLAGRANDE, G. Análise de Domínio: Um Suporte a Análise de Requisitos. Projeto de conclusão de curso. UFSC- Florianópolis, 1993.

NADIN, M. Interface Design and Evaluation - Semiotic implications. in: Hartson, H.R. & Hix, D. (eds.). Advances in Human Computer Interaction. Norwood, New Jersey: Ablex Publishing Corporation, 1988, v. 2.

NEIGHBORS, J. The Draco Approach to Constructing Software from Reusable Components. IEEE Transactions on Software Engineering, September, 1984, pp. 564-574.

NEMES, E. M. & TAGLIARI, P.V. Análise de Linguagens da Aplicação. Projeto de conclusão de curso. UFSC- Florianópolis, 1992.

PEREIRA, R. L. & MORO, R. F. Uma nova experiência no uso da Análise de Domínio como suporte à Análise de Requisitos. Projeto de conclusão de curso. UFSC- Florianópolis,1994.

PRIETO-DÍAZ, R. Domain Analysis for Reusability. COMPSAC'87 Proceedings, Tokyo, Japan, October 1987, pp. 23-29.

SCAPIN, D. The need for a Psycho-engineering approach to HCI. Anais do Segundo Congresso Latino-Americano e Sexto Seminário Brasileiro de Ergonomia. Florainópolis: ABERGO/FUNDACENTRO, 1993.

SCLIAR-CABRAL, L. Curso de Introdução à Lingüística. Pós-Graduação em Lingüística, UFSC, 1993.

TRETER, A. I. & PETRY, P. G.. O Uso do Modelo DOM como Suporte à Análise de Requisitos. Projeto de conclusão de curso. UFSC- Florianópolis,1994.

VALENTIN, A. & outros. L'Évaluation Ergonomique des Logiciels: une démarche itérative de conception. Montrouge: Éditions de l'ANACT, 1993.

WINOGRAD, T. & FLORES, F. Understanding Computers and Cognition: A new foundation for design. (first ed.) Readings, Massachusetts: Addison-Wesley Publishing Company, 1985, 207 p.

Todas as versões de trechos de textos de língua estrangeira foram realizadas pelo Autor.