

**Cecília Kremer Vieira da Cunha**

**Um Modelo Semiótico dos Processos de Comunicação Relacionados à  
Atividade de Extensão à Aplicação por Usuários Finais**

**TESE DE DOUTORADO**

**Departamento de Informática**

**Pontifícia Universidade Católica do Rio de Janeiro**

**Rio de Janeiro, 17 de agosto de 2001.**

**Cecília Kremer Vieira da Cunha**

**UM MODELO SEMIÓTICO DOS PROCESSOS DE COMUNICAÇÃO  
RELACIONADOS À ATIVIDADE DE EXTENSÃO À APLICAÇÃO  
POR USUÁRIOS FINAIS**

Tese apresentada ao Departamento de  
Informática da PUC-Rio como parte dos  
requisitos para a obtenção do título de Doutor em  
Informática: Ciência da Computação

Orientadora: Clarisse Sieckenius de Souza

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 17 de agosto de 2001.

*Dedico esta tese ao Marcus,  
meu incansável companheiro que  
diariamente me ensina sobre a  
incondicionalidade do amor.*

## AGRADECIMENTOS

Uma tese de doutorado depende de uma série de fatores e principalmente pessoas que conspiram a seu favor. Gostaria de agradecer a todos aqueles que me apoiaram durante os anos dedicados a este trabalho.

Gostaria de agradecer ao Marcus, pela sua atenção nos momentos em que compartilhei idéias e sentimentos que faziam parte do meu universo particular e principalmente pela sua enorme compreensão nos momentos em que nos privei de outras coisas para atingir este objetivo.

Gostaria de agradecer à minha família, especialmente minha mãe, meu pai e minhas irmãs, por compreenderem e incentivarem minha dedicação a este trabalho e por me apoiarem nas diferentes fases por que passei durante esses anos.

Gostaria de agradecer aos meus amigos por me manterem motivada e por confiarem em mim.

Gostaria de agradecer especialmente à Clarisse por ter se dedicado à este trabalho, por ter sido compreensiva, paciente e, principalmente, por ter compartilhado seus conhecimentos, experiências e pontos de vista exercendo um papel fundamental na construção do conhecimento de Interação Humano-Computador que tenho hoje.

Gostaria de agradecer a todos os meus amigos do SERG que me ajudaram bastante a desenvolver esse trabalho.

Gostaria de agradecer a todos os meus amigos do TeCGraf por fazerem de lá um ambiente extremamente agradável e enriquecedor para trabalhar.

Gostaria de agradecer a todos os meus amigos do departamento de Informática, especialmente aqueles que compartilharam a hora do almoço comigo e me permitiram amadurecer ainda mais intelectualmente.

Gostaria de agradecer à Judy por ter me adotado no departamento de Technical Communication da Universidade de Washington, por ter me motivado e por ter me dado a oportunidade de enriquecer não só o meu trabalho mas também a minha bagagem cultural. Sou grata também a todos os meus amigos do departamento de Technical Communication e a todos os meus amigos brasileiros em Seattle. Sou especialmente grata à Anna Hester por todo o apoio que ela me deu enquanto estive lá.

Gostaria de agradecer ao CNPq, ao TeCGraf, ao meu pai, ao Marcus e à PUC pelo apoio financeiro sem o qual esse trabalho não teria sido possível.

## Resumo

Aplicações extensíveis por usuários finais representam uma proposta para tratar o problema de que é improvável que um software consiga atender a todas as necessidades específicas de cada usuário em um domínio. Uma das áreas de pesquisa de aplicações extensíveis é a de Programação por Usuários Finais ou *End-User Programming* (EUP).

Investigações empíricas de EUP evidenciam a existência de práticas colaborativas relacionadas ao processo de extensão de aplicação, onde pessoas se comunicam com diversos objetivos: para ajudarem-se a resolver seus problemas com computação, para compartilhar extensões prontas ou construí-las conjuntamente. Nosso trabalho complementa essas investigações, contribuindo com um tratamento aprofundado e teoricamente motivado dos fenômenos de comunicação relacionados às práticas colaborativas observadas. Com base na teoria da Semiótica e na Engenharia Semiótica, propomos um modelo desses fenômenos, descrevendo sua estrutura e comportamento, permitindo assim uma melhor compreensão das questões relacionadas aos mesmos. De acordo com o modelo, projetamos uma linguagem computável para a representação de extensões de forma associada aos discursos das comunicações estudadas.

## **Abstract**

Applications that are extensible by end-users represent a proposal to cope with the issue that it is improbable that a software will attend every specific need of each different user within a domain. One of the research areas approaching extensible applications is End-User Programming (EUP).

EUP empirical investigations evidence the existence of collaborative practices related to the process of application extension. In these practices, people communicate with each other with various purposes: to help themselves deal with their problems regarding computing, to share ready-made extensions or to build extensions together. Our work complements these investigations by offering a deep and theoretically motivated treatment of the communication phenomena related to the observed collaborative practices. Based on Semiotics theory and Semiotic Engineering, we propose a model of these phenomena, describing its structure and behavior thus supporting a better understanding of the issues related to them. According to the model, we designed a computable language for the representation of extensions in association with corresponding communicative discourse.

## Sumário

|  |     |
|--|-----|
| Lista de Figuras.....  | VII |
| Lista de Abreviaturas.....   | IX  |
| 1. Introdução.....   | 1   |
| 2. Fundamentação Teórica.....  | 6   |
| 2.1. Programação por Usuários Finais (EUP).....                                      | 6   |
| Práticas Colaborativas de Programação por Usuários Finais.....                       | 6   |
| Linguagens para Programação por Usuários Finais.....                                 | 10  |
| 2.2. Semiótica e Engenharia Semiótica.....   | 17  |
| Semiótica.....   | 17  |
| Engenharia Semiótica.....  | 18  |
| 3. Um Modelo Semiótico dos Processos Comunicativos Relacionados à Tarefa de EUP..... | 32  |
| 3.1. Estrutura.....  | 33  |
| Agentes Comunicantes: Emissor e Receptor.....  | 33  |
| Canal de Comunicação.....  | 35  |
| Contexto.....  | 36  |
| Mensagem.....  | 41  |
| Códigos de Comunicação.....  | 46  |
| 3.2. Dinâmica.....   | 48  |
| Elaboração de Mensagem.....  | 49  |
| Compreensão de Mensagem.....   | 54  |
| 4. Linguagem de Representação de Discursos Baseada no Modelo (X-DIS).....            | 57  |
| 4.1. Modelo de Domínio.....  | 60  |
| 4.2. Modelo de Interfaces.....   | 72  |
| 4.3. X-DIS: Linguagem de Representação de Discursos.....                             | 76  |
| 5. Avaliação Empírica do Trabalho.....   | 84  |
| 5.1. Preparação e Realização do Teste com Usuários.....                              | 85  |
| Domínio da Aplicação: Empréstimo de Materiais.....                                   | 85  |
| Implementação da Aplicação.....  | 86  |
| Material para Documentação de Extensão.....  | 91  |
| Realização do Teste.....   | 92  |



|   |     |
|---|-----|
| 5.2. Análise dos Resultados .....   | 93  |
| Aplicação do Modelo para Análise das Comunicações sobre Extensões.....          | 94  |
| Agentes Comunicantes.....   | 95  |
| Canal de Comunicação.....   | 98  |
| Contexto .....  | 98  |
| Mensagens.....  | 99  |
| Códigos de Comunicação.....   | 113 |
| Elaboração de Mensagens .....   | 115 |
| Compreensão de Mensagens .....  | 116 |
| Conclusões sobre a Aplicação do Modelo .....                                    | 117 |
| Avaliação Preliminar da X-DIS em relação às Extensões Sugeridas .....           | 117 |
| 6. Discussão .....  | 126 |
| 7. Conclusão e Trabalhos Futuros.....   | 139 |
| 8. Referências.....   | 143 |
| Apêndice I. Implementação da X-DIS.....   | 153 |
| Apêndice II. Propostas de Extensão à Aplicação de Empréstimo de Materiais ..... | 156 |

## Lista de Figuras

|  |    |
|--|----|
| Figura 1. O Princípio da Abstração Interpretativa (com base em [de Souza et al.'2001]) ..... | 21 |
| Figura 2. O Princípio do Contínuo Semiótico (com base em [de Souza et al.'2001]).....        | 24 |
| Figura 3. Continuidade Semiótica entre a UEL, a UIL e a EUPL de um Software Extensível ..... | 26 |
| Figura 4. Extensão do Modelo Semiótico para a Tarefa de EUP .....                            | 33 |
| Figura 5. Sistemas Semióticos Específicos de uma Aplicação Multi-Usuário e Extensível.....   | 43 |
| Figura 6. Sistemas Semióticos Específicos Abordados e a Semiótica da Linguagem Natural ..... | 44 |
| Figura 8. Espiral de Colaboração: Comunicação e Tomada de Decisão .....                      | 49 |
| Figura 9. Elementos de Discurso, A Linguagem Formal e Linguagens Derivadas .....             | 60 |
| Figura 10. Mapa do Modelo de Tarefas.....  | 61 |
| Figura 11. Rede de Atividades do domínio de Empréstimo de Materiais .....                    | 62 |
| Figura 12. Rede de Interfaces da Atividade Empréstimo .....                                  | 63 |
| Figura 13. Exemplo do Diálogo “Preenche Informações do Empréstimo” .....                     | 65 |
| Figura 14. Rede de Interfaces da Atividade Empréstimo .....                                  | 65 |
| Figura 15. Diagrama de Classes do Domínio de Empréstimo de Materiais .....                   | 67 |
| Figura 16. Papéis no Modelo de Grupo da Aplicação de Empréstimo de Materiais .....           | 69 |
| Figura 17. Objetos da Aplicação e Atuação dos Membros do Grupo.....                          | 71 |
| Figura 18. Modelos Previstos .....   | 75 |
| Figura 19. Rede Abstrata de Transição de Estados da Linguagem Formal .....                   | 77 |
| Figura 20. Sentença da X-DIS: Preenchimento de Dados de Empréstimo .....                     | 83 |
| Figura 21. Interface de Informações Adicionais sobre o Locatário.....                        | 86 |
| Figura 22. Interface de Mensagem de Erro de Data Inválida .....                              | 87 |
| Figura 23. Interface de Confirmação de Empréstimo .....                                      | 87 |
| Figura 24. Rede de Interfaces de Devolução de Material .....                                 | 88 |
| Figura 25. Interface de Devolução de Material.....   | 88 |
| Figura 26. Interface de Confirmação de Devolução .....                                       | 89 |
| Figura 27. Interface de Visualização de Vencimentos .....                                    | 90 |

## VIII

|   |     |
|---|-----|
| Figura 28. Rede de Interfaces de Cadastramento de Material .....                          | 90  |
| Figura 29. Interface de Cadastramento de Material.....                                    | 91  |
| Figura 30. Formulário para Documentação de Extensões .....                                | 92  |
| Figura 31. Mapa dos Sistemas Semióticos Previstos.....                                    | 100 |
| Figura 32. Mapa dos Sistemas Semióticos Obtido após o Teste .....                         | 101 |
| Figura 33. Agrupamento das Extensões Sugeridas.....                                       | 113 |
| Figura 34. Exemplo de Uso de Apontadores e Linguagem Natural na Sugestão de Extensão..... | 114 |
| Figura 35. Exemplo de Extensão que Ocasinou uma Abdução Hipocodificada.....               | 116 |
| Figura 36. Matriz Plano de Conteúdo X Plano de Expressão.....                             | 118 |
| Figura 37. X-DIS e Linguagens Derivadas.....  | 121 |
| Figura 38. Edição Visual da Interface de Visualização de Vencimentos .....                | 123 |
| Figura 39. Anotação sobre a Edição da Interface de Visualização de Vencimentos .....      | 124 |

**Lista de Abreviaturas**

|       |  |
|-------|--|
| ATN   | <i>Augmented Transition Network</i>  |
| CSCW  | <i>Computer-Supported Collaborative Work</i>   |
| EUP   | <i>End-User Programming</i> / Programação pelo Usuário Final                         |
| EUPL  | <i>End-User Programming Language</i> / Linguagem de Programação para o Usuário Final |
| HTML  | <i>Hypertext Mark-up Language</i>  |
| LECI  | Linguagem para a Especificação de Cenários de Interação                              |
| X-DIS | Linguagem para Representação de Discursos sobre Sistemas Extensíveis                 |
| UEL   | <i>User Explanation Language</i> / Linguagem de Explicação para o Usuário            |
| UIL   | <i>User Interface Language</i> / Linguagem de Interface com o Usuário                |
| UML   | <i>Unified Modeling Language</i>   |
| WfMC  | <i>Workflow Management Coalition</i>   |
| WWW   | <i>World Wide Web</i>  |

# 1. Introdução

O objetivo central deste trabalho é propor um modelo dos processos de comunicação que estão relacionados às atividades de extensão, por usuários finais, à aplicações de grupo. A partir do modelo, propõe-se uma linguagem artificial para representação dos discursos envolvidos nas atividades de extensão mencionadas. O objetivo é que essa linguagem sirva de base para a construção de aplicações extensíveis que prevêm um apoio aos processos de comunicação sobre as mesmas.

Diversos trabalhos, como por exemplo os de [Suchman'1987 e Myers'1992] explicam porquê é improvável que um software consiga prover soluções para cada problema específico, de cada usuário em particular, em um domínio de atividade. Uma das áreas de pesquisa que se dispõe a tratar especificamente dessa questão é a área de Sistemas de Programação por Usuários Finais ou *End-User Programming* (EUP). Um de seus objetivos é permitir que o próprio usuário estenda a aplicação que utiliza. Diversas soluções foram propostas nesse sentido, tais como a configuração de parâmetros, a gravação de macros, a programação por demonstração e até mesmo o oferecimento de linguagens de programação completas, em que o foco principal reside na interação usuário-sistema [Cypher'1993, Lieberman'2001].

Ainda assim, outros trabalhos de EUP indicam a importância de se tratarem também as questões colaborativas relacionadas ao processo de extensão de aplicação, ou seja, de se tratar a questão da extensão de aplicações com o foco na matriz social que é contexto do uso da aplicação [Mackay'1990, MacLean et al.'1990, Nardi&Miller'1990, Gantt&Nardi'1992, Nardi'1993]. Esses trabalhos são baseados principalmente em estudos empíricos e mostram que comunidades de programação por usuários finais se desenvolvem quando um mesmo programa é utilizado por um grupo de pessoas em um ambiente social, como um escritório ou um laboratório. Essas comunidades surgem espontaneamente com base em redes sociais existentes em que pessoas se comunicam com diversos objetivos: para ajudarem-se a resolver seus problemas com computação, para compensarem lacunas nos seus conhecimentos sobre computadores, para trocarem códigos de programas prontos ou construí-los conjuntamente.

Essa comunicação pode se dar de forma unidirecional ou bidirecional. As comunicações unidirecionais caracterizam principalmente a contribuição espontânea de usuários que disponibilizam extensões prontas para reuso. Nesse caso, estendedores disponibilizam extensões prontas para seus possíveis utilizadores futuros a partir de repositórios de extensão. No caso típico, o estendedor disponibiliza a implementação da extensão em conjunto com uma descrição em linguagem natural sobre a mesma.

Sob a perspectiva da Engenharia Semiótica [de Souza'1993], diz-se também que há, através do software, uma comunicação unidirecional do seu designer para o(s) usuário(s), cuja mensagem transmitida informa quais objetivos podem ser alcançados com o uso da aplicação, de que forma, e quais são as possibilidades de se estender a aplicação. Há também uma comunicação unidirecional entre o usuário (no papel de designer) e ele mesmo (no papel de usuário) quando se realiza uma extensão à aplicação. Nessa comunicação, a mensagem do usuário estendedor trata de objetivos, ou formas de se alcançarem objetivos, que não estão previstos na aplicação original [Silva'2001].

Foram observados também outros tipos de práticas colaborativas relacionadas à atividade de EUP que envolvem comunicações bidirecionais e diretas, ou seja, diálogos, entre os usuários estendedores [Nardi'1993], e até mesmo entre estendedores e designers do sistema [Repenning et al.'1999]. É o caso por exemplo de estendedores mais experientes (ou designers) que auxiliam estendedores menos experientes na construção de uma extensão através de treinamento, de compartilhamento e/ou alteração de códigos existentes e de criação conjunta de extensões. [Mackay'1990] observou também grupos de usuários que concordavam em compartilhar ou criar um conjunto de extensões de forma a se comprometerem conjuntamente com uma determinada prática de trabalho. Essa observação ocorreu no âmbito de um sistema de envio de e-mail que permitia aos usuários construir extensões para filtragem e organização de mensagens.

Ao aplicarmos os conceitos de EUP a sistemas de grupo verificamos a existência de um tipo de colaboração similar. Suponha-se um sistema que auxilia um grupo de usuários a realizar conjuntamente uma tarefa para atingir um objetivo comum, em que diferentes membros do grupo são responsáveis por passos distintos da mesma tarefa, como é o caso típico de sistemas

de *workflow*. Ao se construir uma extensão para essa tarefa (mantendo-se os mesmos passos originais e, por exemplo, modificando seus dados), é preciso que o estendedor adquira, pelo menos, um comprometimento dos outros membros do grupo sobre a utilização futura da extensão, ou até mesmo a ajuda deles para tornar a construção da extensão possível. Caso contrário, a extensão pode se tornar irrealizável ou inútil. Nessa situação, a colaboração entre usuários durante a extensão é não só desejável, mas imprescindível para a eficácia da extensão. A colaboração envolverá necessariamente uma negociação entre o estendedor e os usuários que serão afetados pela nova extensão. A situação é crítica quando há uma relação hierárquica entre os usuários do grupo, e o estendedor se encontra em nível hierárquico inferior ou equivalente em relação aos outros usuários afetados pela extensão.

A situação exemplificada acima mostra um caso em que o apoio à bidirecionalidade de comunicação é imprescindível. Em outra dimensão, a bidirecionalidade é fator crucial para permitir um entendimento mútuo entre os agentes comunicantes. É o que acontece quando os agentes não estão de acordo com o significado dado a uma expressão e precisam negociá-lo, fazendo uso para isso de recursos metalingüísticos. Por exemplo, quando um agente pergunta a outro: “O que você quer dizer com X?” ou “X significa Y?” ele está na verdade tentando atribuir um significado à expressão X que foi utilizada pelo outro agente em mensagem anterior, mas não foi compreendida. Esse tipo de negociação deve estar prevista em modelos de processos comunicativos de EUP, tendo em vista que a tarefa de extensão apresenta a característica peculiar de que, através dela, os estendedores irão (potencialmente) construir e comunicar significados novos, que não existem na aplicação original e portanto não são necessariamente conhecidos pelos outros agentes comunicantes.

Apesar dessas observações empíricas e das argumentações de que aplicações normalmente não conseguem atender a todas as necessidades de todos os usuários, [MacLean et al.'1990] mostraram em 1990 que não tínhamos atingido ainda uma cultura de adaptação de sistemas, na qual os usuários teriam um senso de propriedade e controle sobre o sistema que utilizam e a construção de adaptações seria a regra (e não a exceção). Afirmações recentes ainda corroboram essa perspectiva ao declararem que até o momento não há uma adoção difundida de tecnologias de EUP [Shneiderman'2001, Lieberman'2001].

Como caminhos para se alcançar a cultura de adaptação, [MacLean et al.'1990] citam a construção de mecanismos de adaptação mais acessíveis e o tratamento da questão de adaptação como um esforço comunitário. A maioria das pesquisas de EUP tem se concentrado no primeiro caminho e por conseguinte há hoje uma grande variedade de soluções de linguagens e mecanismos de EUP. Esse trabalho procura trilhar o segundo caminho.

Se utilizarmos como definição de cultura a de “um conjunto de pensamentos que são compartilhados entre membros de um grupo” [Levine&Moreland'1991], percebemos que a criação de uma cultura de adaptação depende fortemente de processos comunicativos, pois é através deles que agentes comunicantes acumulam uma base de conhecimento compartilhada (*common ground*) [Clark&Brennan'1991] sobre, por exemplo, o que é adaptação e as formas de realizá-la.

No entanto, as pesquisas de EUP ainda não se voltaram para um tratamento aprofundado e teoricamente motivado do fenômeno de comunicação envolvido nas práticas colaborativas observadas empiricamente. Essa é uma das contribuições que pretendemos fornecer com esse trabalho, já que apresentamos um modelo desse fenômeno de comunicação tomando como base a teoria Semiótica, que trata dos processos de comunicação e significação em geral, e a Engenharia Semiótica, que aplica essa teoria aos fenômenos de Interação Humano-Computador.

Nosso modelo caracteriza o fenômeno de comunicação sobre extensões, descrevendo sua estrutura e comportamento e permitindo uma melhor compreensão das questões relacionadas ao mesmo. Logo, podemos dizer que esse trabalho oferece uma base para futuros métodos e ferramentas que pretendam atuar como facilitadores dos processos e atividades em que os usuários se envolvem ao estender uma aplicação.

De acordo com o modelo, projetamos uma linguagem artificial e computável para a representação de discursos envolvidos na atividade de extensão, de maneira que as comunicações possam ocorrer de forma situada e integrada a essa atividade. Essa linguagem reflete as características fundamentais das comunicações sobre extensões. Primeiro temos que a natureza do objeto do discurso (a extensão) é uma aplicação computacional interativa e que,



a princípio, quer-se representar esse objeto nos mesmos termos da aplicação que o originou, ou seja, de forma computável e interativa. Daí a necessidade de prover-se ao menos uma linguagem formal e computável para representação desse objeto. Segundo, é inerente ao objeto do discurso um caráter de novidade em relação à aplicação original. [Eco'1976] descreve que a criação de novos significados se dá a partir de significados existentes em um processo de oposição (no sentido de segmentação). Assim, é importante que o discurso expresse a nova segmentação proposta pela extensão de forma relacionada à segmentação da aplicação original, destacando assim, para um receptor, a novidade proposta pela extensão comunicada. Assim, temos que essa linguagem diferencia-se das linguagens e mecanismos de representação de extensão atuais, no sentido em que adota uma finalidade primordialmente comunicativa.

Em linhas gerais podemos dizer também que, ao permitir a construção de discursos a partir de uma linguagem como esta, estamos colaborando para um aumento no potencial de uso do computador como mídia [Kammersgaard'1988, Andersen et al.'1993], tendo-se que este aspecto permanece sub-explorado pela população de usuários não especialistas em computação, especialmente no que tange à possibilidade de expressão de mensagens interativas, dados os conhecimentos de programação que a construção dessas mensagens requer [diSessa&Abelson'1986].

No Capítulo 2, apresentaremos o contexto dessa pesquisa e os principais conceitos aplicados ao longo desse trabalho. O Capítulo 3 descreve o modelo proposto detalhando as visões estruturais e comportamentais do fenômeno de comunicação sobre extensões. O Capítulo 4 apresenta a linguagem artificial de representação dos discursos previstos no âmbito desse trabalho e os elementos de aplicação que são integrados através dela. O Capítulo 5 discorre sobre a investigação empírica realizada como parte da avaliação desse trabalho e o Capítulo 6 complementa essa avaliação através de uma comparação desse trabalho com seus co-relatos. Finalmente no Capítulo 7 consolidamos nossas contribuições e apontamos algumas direções para trabalhos futuros. Em anexo encontra-se o código em Prolog da implementação da linguagem de representação de discursos.

## 2. Fundamentação Teórica

Neste capítulo apresentamos os principais referenciais deste trabalho e descrevemos os conceitos que serão aplicados posteriormente na descrição do modelo e da linguagem de representação de discurso propostos.

### 2.1. Programação por Usuários Finais (EUP)

As pesquisas em Programação por Usuários Finais ou *End-User Programming* (EUP), juntamente com as pesquisas da Engenharia Semiótica, constituem o principal foco da contribuição desse trabalho. Pesquisas de EUP indicam a importância de se tratarem as questões colaborativas relacionadas ao processo de construção de extensões. Baseadas principalmente em estudos empíricos, elas mostram que comunidades de programação por usuários finais se desenvolvem quando um mesmo programa é utilizado por grupo de pessoas em um ambiente social, como um escritório ou um laboratório. Essas comunidades surgem espontaneamente, com base em redes sociais existentes, em que pessoas procuram umas às outras com diversos objetivos: para ajudarem-se a resolver seus problemas com computação, para compensarem lacunas nos seus conhecimentos sobre computadores, para trocarem códigos de programas prontos ou construí-los conjuntamente.

No entanto, as pesquisas de EUP ainda não se voltaram para um tratamento aprofundado e teoricamente motivado do fenômeno de comunicação envolvido nessas colaborações. É essa contribuição que pretendemos fornecer com esse trabalho através de uma abordagem que aplica a teoria Semiótica e Engenharia Semiótica. Portanto, de forma a compreender o cenário no qual esse trabalho se insere, fazemos uma revisão dos resultados das pesquisas de EUP que abordam as questões colaborativas.

#### Práticas Colaborativas de Programação por Usuários Finais

No trabalho de [MacLean et al.'1990] argumenta-se que não é necessário ter somente sistemas que possam ser adaptados pelos usuários finais, mas também uma cultura de

adaptação de sistemas. Nessa cultura os usuários teriam um senso de propriedade e controle sobre o sistema que utilizam e a construção de adaptações seria a regra (e não a exceção). Afirmações recentes confirmam que não atingimos ainda o estágio da cultura de adaptação, levando-se em conta que ainda não há adoção difundida das tecnologias de EUP [Shneiderman'2001, Lieberman'2001].

Em [MacLean et al.'1990] os autores caracterizam três culturas que concentram pessoas com níveis diferentes e crescentes de conhecimento sobre adaptação:

- A cultura do trabalhador: é a cultura na qual, em linhas gerais, se concentram as pessoas que não têm interesse em sistemas de computador em si. O foco está em realizar o trabalho e não há expectativas de adaptar o sistema;
- A cultura do explorador: na qual se concentram os trabalhadores que gostam de explorar os sistemas computacionais, embora possam não entendê-los completamente; e
- A cultura do programador: na qual se concentram pessoas que têm um treinamento formal ou extensa experiência em computação. São essas pessoas que podem conhecer e entender completamente um sistema, exercendo o papel de suporte a uma aplicação. Normalmente uma pessoa com esse perfil não é acessível aos trabalhadores.

Segundo os autores, um dos desafios para a criação de uma cultura integrada de adaptação de sistemas é a cultura do trabalhador, na qual não há expectativas de dominarem-se as adaptações que são feitas a um sistema e onde, portanto, não se procura entender quais adaptações seriam possíveis e nem como elas poderiam ser realizadas. Por conseguinte, a comunicação dos usuários dessa cultura com aqueles que poderiam vir a construir adaptações, sobre suas novas idéias e requisitos, tende a se tornar problemática.

Logo, o objetivo é dar aos trabalhadores a sensação de propriedade do sistema e de domínio para realizar e entender quais adaptações podem ser feitas. Assim, pode-se esperar que usuários efetuem adaptações, saibam quem na comunidade pode ajudá-los e saibam o que pedir e como interpretar as ofertas de adaptação que lhes são feitas por outros.

Há dois caminhos para se alcançar esse objetivo [MacLean et al.'1990]. O primeiro é o de tornar os mecanismos de adaptação mais acessíveis, diminuindo, e tornando mais linear e gradativo o esforço de aprendizado exigido de uma pessoa na medida em que ela obtém poder de adaptação de um sistema. Isso permitiria uma diminuição na discrepância existente entre as três culturas mencionadas. O outro caminho é o de tratar-se a questão de adaptação como um esforço comunitário. O objetivo é formar uma cultura única, na qual os diferentes e importantes papéis dos programadores, exploradores, e trabalhadores sejam reconhecidos de maneira a beneficiar a comunidade como um todo. Mais uma vez, a diminuição da discrepância existente entre as três culturas, proposta como primeiro objetivo, deve permitir que pessoas com conhecimentos diferentes tenham um contato maior e se comuniquem melhor umas com as outras.

[MacLean et al.'1990] destacam um papel, surgido a partir de uma experiência de projeto, que deve ser adicionado aos já descritos: o de faz-tudo. A pessoa exercendo esse papel é aquela que une trabalhadores, exploradores e programadores. Ela atua junto à equipe de trabalho e responde às suas necessidades imediatas. E ela é também capaz de comunicar as necessidades dos usuários a programadores para desenvolvimentos mais complexos ou de longo prazo.

Os comportamentos associados a todos esses diferentes perfis foram observados em pesquisas de EUP em diversos domínios [Clement'1990, Mackay'1990, Nardi&Miller'1990, Gantt&Nardi'1992].

Ao analisarmos essas observações segundo uma definição de cultura como “um conjunto de pensamentos que são compartilhados entre membros de um grupo” [Levine&Moreland'1991], percebemos que a criação de uma cultura de adaptação depende fortemente de processos comunicativos, pois é através deles que se cria uma base de conhecimento compartilhada [Clark&Brennan'1991] entre um grupo de pessoas sobre, por exemplo, o que é adaptação e as formas de realizá-la.

Note-se que, nas três culturas mencionadas, as bases de conhecimento independentes, pré-existentes, especialmente as de trabalhadores e programadores, tendem a ter pouca interseção. Isso dificulta um processo comunicativo, já que ele constitui uma evolução sobre um

conhecimento que já é partilhado entre os agentes comunicantes. Logo, explica-se a importância dada ao papel de faz-tudo nos trabalhos sobre EUP colaborativo. É o faz-tudo que, por ter conhecimento das três culturas diferentes, pode ajudar na construção da base de conhecimento comum ao grupo. Por exemplo, ele utiliza seus conhecimentos sobre o domínio da aplicação para comunicar aos trabalhadores conhecimentos sobre computação (e.g. como realizar determinada tarefa no sistema) e utiliza seus conhecimentos sobre computação para comunicar aos programadores conhecimento sobre o domínio (e.g. os requisitos dos trabalhadores).

[Mackay'1990] apresenta, como conceito análogo ao de faz-tudo, o papel de intérprete no sentido em que o intérprete é aquele que oferece ajuda aos usuários que têm menos conhecimento sobre os mecanismos de adaptação do sistema para que eles atinjam seus objetivos individuais. Seguindo a analogia, poderíamos dizer que cada uma das três culturas identificadas “fala uma língua diferente” e que o intérprete atua como um intérprete para pessoas de diferentes culturas. Podemos dizer também que a área de EUP atua no sentido de ensinar aos usuários finais a linguagem dos programadores.

Nesse sentido, [MacLean et al.'1990] dizem que o primeiro passo para criar-se uma cultura de adaptação é um ambiente que seja tão fácil de adaptar quanto de usar, pelo menos no que tange a algumas adaptações. Assim, motiva-se o usuário a fazer uma primeira tentativa de adaptação. Depois, essa motivação deve ser mantida ao longo de adaptações gradativamente mais complexas e mais poderosas.

Essa proposta gradativa pode ser vista como uma forma de aprendizado da linguagem dos programadores, pelos usuários, a qual se inicia com uma linguagem já conhecida (e.g. a da interface) e gradativamente caminha em direção a uma linguagem de programação (e.g. a linguagem LISP, utilizada no exemplo de [MacLean et al.'1990]). Note-se que é parte importante dessa proposta a disponibilidade de ajuda por parte de outra pessoa (e.g. o faz-tudo ou o intérprete). Cabe notar que essa ajuda dada pelo intérprete é representada não só por ensinamentos e treinamento, mas também pela distribuição direta de códigos prontos e modificação de códigos existentes [Mackay'1990].

A maioria das pesquisas de EUP tem se concentrado nas diferentes linguagens que podem ser fornecidas aos usuários dentro do espectro de gradação mencionado anteriormente. Fazemos então uma revisão dos tipos de solução de linguagens de EUP já propostos. Assim poderemos verificar quais são, atualmente, as principais faixas e recursos que os usuários têm para expressão de extensões e seus custos de aprendizado.

## **Linguagens para Programação por Usuários Finais**

Nossa revisão utiliza como ponto de partida a classificação de paradigmas de EUP descrita por [Silva'2001], já que a mesma emprega a Engenharia Semiótica como base teórica de análise. A classificação diferencia cada paradigma pelo tipo de linguagem empregada e suas possibilidades de extensão. Descrevem-se os seguintes paradigmas principais de programação por usuários finais: programação paramétrica, imitativa e descritiva. Tipicamente essa seqüência de paradigmas representa uma ordem crescente de expressividade da linguagem, assim como custo de aprendizado pelo usuário final.

### **Programação Paramétrica**

A programação paramétrica permite que um estendedor modifique uma aplicação a partir da atualização de valores de parâmetros disponibilizados. Existem dois níveis possíveis de atuação do usuário na extensão a uma aplicação neste paradigma. No primeiro, o usuário está limitado a selecionar entre alternativas e a mudar os valores para os parâmetros disponibilizados pelo designer da aplicação extensível. Assim, diz-se normalmente que um usuário está configurando a aplicação.

No segundo nível de atuação, um usuário pode atribuir um tipo de identificação (e.g. um nome e/ou um ícone) a uma combinação de diferentes parâmetros (com seus respectivos valores). A identificação é então adicionada à interface da aplicação como um novo item léxico e, ao ser referenciada, ativa a combinação de parâmetros e valores associada a ela.

Um exemplo deste paradigma é a criação de estilos no editor de textos *MS Word*<sup>TM</sup> [Silva'2001]. Pode-se especificar por exemplo um tipo de formatação de parágrafos de um

texto através do agrupamento de um conjunto de valores para parâmetros que determinam: o alinhamento dos parágrafos, o estilo e tamanho do fonte dos caracteres, etc. Depois de especificado, o tipo de formatação é associado a uma nomenclatura (e.g. Bibliografia) que passa a pertencer ao léxico da interface do editor de texto.

Esse paradigma apresenta um custo de aprendizado baixo, desde que não se eleve muito a quantidade de parâmetros e valores alternativos associados. Em decorrência disso, ele apresenta também uma baixa expressividade.

### **Programação Imitativa**

O segundo paradigma de EUP é o da programação imitativa, que abrange dois mecanismos principais: a gravação de macros e a programação por exemplo ou por demonstração [Cypher'1993, Lieberman'2001]. Na gravação de macros um usuário dispõe de mecanismos para criar réplicas de seqüências de passos interação. Basicamente, o usuário ativa um “gravador” e depois interage normalmente com a aplicação, mostrando a seqüência de interações a ser gravada. Ao término da gravação o usuário associa uma identificação à seqüência gravada (e.g. um item de menu ou tecla de atalho), que assim passa a poder ser invocada e reproduzida automaticamente.

Note-se que, enquanto no paradigma anterior o usuário atuava apenas no léxico da linguagem da interface original, nesse paradigma ele passa a atuar também no seu nível sintático. Através desse tipo de programação o usuário consegue criar seqüências de passos de interação, ou seja, novas sentenças dentro da gramática da linguagem de interface. Isso indica um aumento no poder de expressão do usuário para descrever suas extensões, uma vez que torna-se possível, a partir do conjunto fixo de elementos lexicais da interface, obter um conjunto infinito de seqüências de interação ou sentenças da linguagem de interface.

A maior vantagem da gravação de macros é a familiaridade do usuário com a linguagem usada na especificação da extensão, que é a própria linguagem de interface da aplicação. Por outro lado, as gravações normalmente armazenam os passos interativos de forma literal e dependente de contexto. Por exemplo, suponha que um usuário está editando um arquivo

chamado TESE e grava uma macro que o salva em formato HTML. Suponha agora que o usuário está editando um arquivo chamado CARTA e deseja utilizar a macro gravada para salvá-lo em formato HTML. Depois de invocar a macro, o usuário percebe que seu arquivo foi gravado como TESE.HTML. Isso é o efeito de uma restrição do mecanismo de gravação de macros que não consegue identificar pontos onde construtos mais abstratos (e.g. variáveis, condições e iterações) deveriam ser inseridos.

Uma das soluções propostas a esse problema é o mecanismo de programação por demonstração, chamado também de programação por exemplos. Ele utiliza técnicas de inteligência artificial que, a partir de um conjunto de diferentes exemplos de passos de interação mostrados pelo usuário, criam programas mais generalizados que podem ser utilizados em diferentes situações [Cypher'1993, Myers'2000, Lieberman'2001].

Essa solução mantém a vantagem do mecanismo de gravação de macros: a familiaridade do usuário com a linguagem de extensão. Adiciona-se também um aumento no poder de expressão da linguagem através da possibilidade de inferência dos construtos mais abstratos de programação, lembrando que o usuário não precisa aprendê-los diretamente.

Por outro lado, para poder fazer o sistema reconhecer sua real intenção e conseqüentemente gerar as extensões desejadas, um usuário precisaria conhecer as inferências realizadas pelo sistema, que normalmente não lhe são apresentadas. Portanto, embora essa solução não exija conhecimentos de programação do usuário e capitalize sobre seu conhecimento sobre a linguagem de interface, ela exige um esforço cognitivo do usuário em relação aos mecanismos de inferência subjacentes.

Além da programação por exemplo, outra forma de lidar com o problema da falta de abstração do processo de gravação de macros é deslocar a responsabilidade das especificações mais abstratas para o usuário. É o caso por exemplo da edição do texto de um programa gerado a partir da gravação de uma macro. Nesse caso há uma mudança do paradigma de programação imitativa para descritiva.



## Programação Descritiva

Na edição de macros o usuário passa a ter acesso direto à linguagem de extensão textual em que as macros são armazenadas. Isso exige uma carga de aprendizado do usuário, que precisará lidar com uma nova linguagem. O aprendizado é dificultado pelo fato de que não há normalmente uma co-relação ou co-referência [Draper'1986] alta entre a linguagem de interface e a linguagem textual. Ou seja, é difícil reconhecer quais elementos da linguagem de interface se referem a quais elementos da linguagem textual. Assim exige-se um grande esforço de usuários que tentem fazer um mapeamento entre a linguagem de interface que já é conhecida e suas referências na linguagem de extensão a ser aprendida.

Para lidar com esse problema, propôs-se o mecanismo de aprendizagem por desvelamento ou revelação [Eisenberg'1994, DiGiano&Eisenberg'1995, Eisenberg'1995]. A proposta básica é a de mostrar ao usuário, a cada passo da sua interação com a interface da aplicação, qual seria o referente (e.g. o comando) correspondente na linguagem textual de extensão. Assim, este mecanismo fornece ao usuário uma forma mais suave de aprendizado da linguagem de extensão.

Uma alternativa às linguagens textuais são as linguagens de programação visual. Elas tentam reduzir a complexidade da tarefa de programação por meio da disponibilização de representações primordialmente visuais, e não lingüísticas, dos construtos de uma linguagem de programação. As linguagens visuais disponibilizam, por exemplo, ícones para representação de dados e operações e diagramas de Venn para descrição de conjuntos de dados e suas relações. Elas buscam capitalizar sobre a habilidade de processamento de informações visuais dos seres humanos, na qual a percepção de relações espaciais leva à inferência de significados sobre essas relações (e.g. **seqüência** de comandos).

A análise feita em [Nardi'1993] mostra que essas características são mais ou menos vantajosas de acordo com a tarefa de programação que o usuário está realizando e seu estilo ou estratégia de programação. As representações visuais são mais adequadas a uma visão geral e estrutural de um programa e sua relação com outros programas. Esse é o caso por exemplo de visualização de programas concorrentes e programas para a criação de interfaces

gráficas. Por outro lado, representações textuais e lingüísticas são melhores para tarefas como a descrição detalhada de comandos e anotações. Assim, conclui-se que a usabilidade de um ambiente visual de programação para usuários finais depende da sua combinação com estruturas lingüísticas textuais [Rader et al.'1998].

É nesse sentido que [Nardi'1993] defende a utilização de uma linguagem de programação visual híbrida, através da qual usuários utilizam formalismos visuais ou *frameworks* visuais de aplicação que se articulam com linguagens de programação textuais. Como exemplo a autora cita a ferramenta ACE (*Application Construction Environment*) [Zarmer&Chew'1992, Zarmer et al.'1992, Johnson et al.'1993] que possui três camadas:

- ACEKit: é uma biblioteca de classes que provê informações sobre objetos que poderão ser manipulados através de editores. Ela é utilizada por programadores para construir, por exemplo, os objetos do domínio da aplicação que farão parte de um ambiente especializado de construção de aplicações para os usuários finais;
- Formalismos Visuais: são *frameworks*, também construídos a partir do ACEKit, para edição visual. Cada tipo de formalismo visual possui uma semântica bem definida para expressar relações. Alguns exemplos são: tabelas, grafos e mapas. Eles são utilizados interativamente por usuários finais, provendo uma base para a organização de uma aplicação como um todo; e
- Linguagens de extensão: são linguagens textuais específicas de tarefa com operações mapeadas diretamente para as tarefas dos usuários. Um exemplo é a linguagem de fórmulas utilizada em planilhas eletrônicas.

Propõe-se que os usuários finais construam a maior parte da aplicação tomando como base os ambientes extensíveis providos pelos programadores. A construção da aplicação se dá basicamente a partir do acoplamento de objetos da biblioteca de classes a elementos de um formalismo, e da descrição de comandos na linguagem de extensão.

Essa forma de programação por usuários finais exige que usuários aprendam uma nova linguagem que é mais complexa do que as linguagens de outros mecanismos, mas também mais expressiva. Diferentemente das linguagens textuais, os usuários podem chegar a

construir aplicações sem conhecer conceitos abstratos de programação. Cabe notar que a idéia é que um usuário construa, a partir dos elementos disponibilizados pelo programador, uma aplicação nova. Ou seja, não há necessariamente uma aplicação original a ser estendida, mas um conjunto de blocos de construção a serem combinados pelo usuário.

Para concluir a seção de Programação por Usuários Finais teceremos algumas considerações sobre os mecanismos descritos a partir de uma perspectiva comunicativa, no sentido de apoiar-se uma cultura de adaptação. Vimos no início da seção que o perfil do trabalhador representa um desafio à criação dessa cultura devido à sua dificuldade de comunicação com pessoas de outros perfis que têm maior conhecimento de programação. Ao falarmos de comunicações sobre extensões, principalmente aquelas que visam uma construção colaborativa da extensão, é preciso perceber que a familiaridade com a linguagem na qual uma extensão é expressa deve ser não só do estendedor (do emissor da mensagem) mas também do co-estendedor ou do usuário da extensão (o receptor da mensagem). Cabe lembrar que essa linguagem de expressão deve ser também processável por computador, de forma que a extensão possa ser efetivada e integrada à aplicação original.

As descrições dos mecanismos de EUP existentes mostraram diferentes formas de expressão de extensão. Elas mostraram também que quanto mais esses mecanismos se aproximavam da linguagem de interface de uso da aplicação, menor era o custo de aprendizado associado ao mecanismo. Cabe notar que isso corrobora o fato de que linguagem de interface de uso da aplicação normalmente é construída de maneira a prever as diferentes culturas, inclusive a dos trabalhadores. Assim podemos dizer que uma aplicação original e sua linguagem de uso constituem uma base de conhecimento que é compartilhada entre as pessoas das diversas culturas. Ou seja, há de fato uma cultura de uso de uma aplicação que pode ser utilizada como ponto de partida para o desenvolvimento da cultura de adaptação.

Por todos esses fatores podemos concluir que, em geral, a linguagem de interface da aplicação é a linguagem artificial computável ideal para representação de extensões nos processos de comunicação que viabilizarão e desenvolverão a cultura de adaptação, especialmente os processos de comunicação que envolvem trabalhadores.

Nos processos de comunicação onde o receptor irá apenas utilizar uma extensão já construída, temos que essa conclusão gera o requisito de que a extensão apresente algum tipo de representação na interface da aplicação, como uma identificação que permita a sua invocação (e.g. nome de item de menu) e uma amostragem dos efeitos de sua execução, de forma que o receptor perceba que a extensão foi executada.

Já nos processos de comunicação que envolvem uma construção colaborativa da extensão, poderia ser necessário que o receptor da mensagem entendesse também a linguagem de extensão. Para incluir-se a cultura do trabalhador nesses processos, o ideal é que a extensão seja expressa ao menos de forma relacionada à linguagem de interface da aplicação original, de forma que o receptor possa se beneficiar do conhecimento que já possui para entender a mensagem sobre a extensão a ser construída. Nesse sentido, os mecanismos de EUP descritos anteriormente apresentam diversas abordagens em um gradiente de carga cognitiva e poder de expressão.

Ainda assim, em todos os mecanismos apresentados há a possibilidade de realizarem-se extensões que destruam a interface original da aplicação. É o caso por exemplo de um usuário que redefina completamente os menus, itens de menus e ícones da barra de ferramenta da aplicação. Lembrando que criação da cultura de adaptação e os processos comunicativos dependem de uma base de conhecimento compartilhada, não é interessante deixar que a interface original seja perdida. Logo deve-se manter, mesmo depois de processos de extensão, ao menos uma versão da interface original que seja acessível para fins de consulta e referência pelos diferentes agentes comunicantes.

Assim finalizamos as considerações sobre a relação entre os mecanismos de EUP descritos e as questões de desenvolvimento de uma cultura de adaptação. Na próxima seção descreveremos alguns conceitos básicos da Semiótica e da Engenharia Semiótica que são aplicados ao longo desse trabalho.

## 2.2. Semiótica e Engenharia Semiótica

### Semiótica

A Semiótica [Eco'1976, Peirce'1931] é base fundamental para nossa proposta, já que teoriza sobre os processos de comunicação e significação. Um conceito central à Semiótica é o de signo. Um signo é algo que está para alguma coisa, que possui significado, para alguém [Peirce'1931]. Ele é o produto de uma relação entre um *representamen* (que representa alguma coisa para alguém, por exemplo a palavra: maçã), um objeto (a coisa que ele representa, por exemplo o objeto real maçã) e seu interpretante (um pensamento, uma sensação, uma ação ou um outro signo, por exemplo a imagem mental de uma maçã ou a sensação de fome) [Nadin'1988]. O interpretante de um signo pode ser um outro signo que, por sua vez, será interpretado em um número indefinido de camadas de significados, trazendo à mente uma variedade de outros signos e significados. Por exemplo, a imagem mental da maçã poderia ser interpretada como um signo de pecado e assim por diante. Este processo de geração de uma cadeia de significados não tem limites por natureza e é denominado de semiose ilimitada.

Durante um processo de comunicação, emissor e receptor trocam mensagens constituídas de um conjunto de signos. Eles restringem o processo de semiose ilimitada de forma pragmática [Nadin'1988], buscando alcançar uma configuração estável na qual entendem que seus interpretantes sobre a mensagem comunicada são compatíveis entre si. Até atingir esse estágio os agentes comunicantes permanecem em conversação, trocando mais mensagens nas quais negociam seus interpretantes. Esses processos de negociação são minimizados na medida em que o emissor expressa sua mensagem utilizando signos que fazem parte de um código estável e compartilhado com o receptor.

Um código é definido como uma regra que associa alguns elementos de um sistema sintático a elementos de um sistema semântico ou de uma série de possíveis respostas comportamentais por parte do receptor [Eco'1976]. Ou seja, um código associa formalmente e sistematicamente elementos segmentados do plano de expressão a elementos segmentados do plano de conteúdo. Assim, “o código estabelece tipos gerais e produz a regra que gerará *tokens* ou

ocorrências concretas (unidades que se realizam no processo comunicativo e que comumente se chamam signos)” [Eco’1976]. Nesse trabalho estaremos utilizando a palavra linguagem como sinônima de código.

Basicamente em uma comunicação, um emissor envia uma mensagem para um receptor através de um meio. Um meio pode aceitar uma faixa de específica de formas ou sinais. Assim, o emissor deve escolher um código adequado ao meio de transmissão ou um meio de transmissão adequado ao código que ele deseja utilizar. Ao utilizar um código, o emissor escolhe, dentre os signos disponíveis nesse código, aqueles que transmitam as intenções e significados desejados e espera que o receptor da mensagem os decodifique e interprete da mesma maneira [Jakobson’1960].

### **Engenharia Semiótica**

A Engenharia Semiótica vem aplicando essa teoria em pesquisas que tratam dos fenômenos de Interação Humano-Computador, dentre eles o uso do software extensível. Este é visto como um fenômeno de comunicação, no qual prevê-se duas situações distintas [de Souza’1993, de Souza et al.’2001]:

1. A de uso do software pelo usuário para a realização de uma tarefa, onde se dá uma comunicação unidirecional do designer para o usuário, cuja mensagem transmitida responde às seguintes questões fundamentais:
  - qual é a interpretação do designer sobre os objetivos do usuário, ou seja, qual é o modelo de funcionalidade da aplicação,
  - como os usuários deveriam interagir com a aplicação para atingir esses objetivos, ou seja, qual é o modelo de interação da aplicação, e
  - quais são as possibilidades de se estender a aplicação, ou seja, qual é o seu modelo de extensibilidade.
2. A de uso do software na criação de uma extensão, onde, além da comunicação designer-usuário, há também uma comunicação unidirecional entre um usuário (no papel de designer) e ele mesmo (no papel de usuário) ou outros usuários [Silva’2001]. Nessa

comunicação, a mensagem do usuário estendedor trata de objetivos, ou formas de se alcançar objetivos, que não estão previstos na aplicação original.

Nessas comunicações unidirecionais de uma aplicação extensível, as mensagens transmitidas são codificadas em um ou mais dos seguintes códigos:

- a UIL (*User Interface Language* ou Linguagem de Interface com o Usuário). Esta é a linguagem para uso normal da aplicação. Como explicado em [Leite'1998] a mensagem do designer é composta por signos produzidos pela articulação de diversos códigos, tais como frases da linguagem do usuário, códigos gráficos, cores, etc. Ela constitui um **texto** [Eco'1976] no sentido em que sua estrutura é o resultado da coexistência de vários códigos. No escopo desse trabalho, para efeitos de simplificação de leitura, estaremos falando dos códigos artificiais de forma geral, abstraindo do fato de que mensagens construídas de acordo com eles constituem de fato um produto de extra-codificação, ou seja de construção de textos no sentido definido por [Eco'1976];
- a EUPL (*End User Programming Language* ou Linguagem de Programação para o Usuário Final). Esta é a linguagem para construção de extensões à aplicação. A subseção sobre Linguagens de Programação por Usuários Finais fez uma revisão de diferentes tipos de EUPL. ; e

- a UEL (*User Explanation Language*). Esta é a linguagem para explicação da aplicação (e.g. manuais *on-line*) e, possivelmente, suas extensões. Diferentemente de alguns trabalhos como os de [de Souza'1997 e Silva'2001], a maioria das linguagens ou ambientes de EUPL não prevêem um apoio integrado para a construção de explicações sobre extensões.

A Engenharia Semiótica trata também dos relacionamentos que devem existir entre esses códigos. Como vimos na sub-seção de Linguagens de Programação por Usuários Finais, um dos problemas de aprendizado dessas linguagens era decorrente da falta de co-referencialidade [Draper'1986] entre elas e a linguagem de interface, o que ocasionava dificuldades para um usuário mapear seu conhecimento sobre a UIL para construtos da EUPL. A Engenharia Semiótica vai além do conceito de co-referencialidade e propõe princípios reguladores dos relacionamentos entre esses códigos [de Souza et al.'2001, Silva'2001]. Esses princípios são denominados Princípio da Abstração Interpretativa e Princípio do Contínuo Semiótico. Para descrever esses princípios precisamos entender que em um código ou linguagem distingue-se os seguintes tipos de signo [de Souza et al.'2001, Silva'2001]:

- um **signo lexical** é uma palavra significativa desta linguagem;
- um **signo frasal** é um construto gramatical válido desta linguagem, ou seja, uma organização estruturada de signos lexicais que contém um significado completo;
- um **signo realizado** em uma linguagem é uma palavra ou sentença existente nesta linguagem; e
- um **signo potencial** em uma linguagem é uma palavra ou sentença inexistente nesta linguagem, mas que pode ser gerado por extensões em seu vocabulário ou em sua gramática, extensões estas que obedecem aos padrões derivacionais desta linguagem.

A seguir descrevemos esses princípios com base em [de Souza et al.'2001 e Silva'2001].



## Princípio da Abstração Interpretativa

O princípio da Abstração Interpretativa procura avaliar a abstração que um código computacional faz daqueles sobre os quais ele está implementado, tomando como ponto de vista a sua interpretação pelo usuário final. Vejamos sua definição formal [de Souza et al.'2001, Silva'2001] a partir do esquema correspondente mostrado na Figura 1:

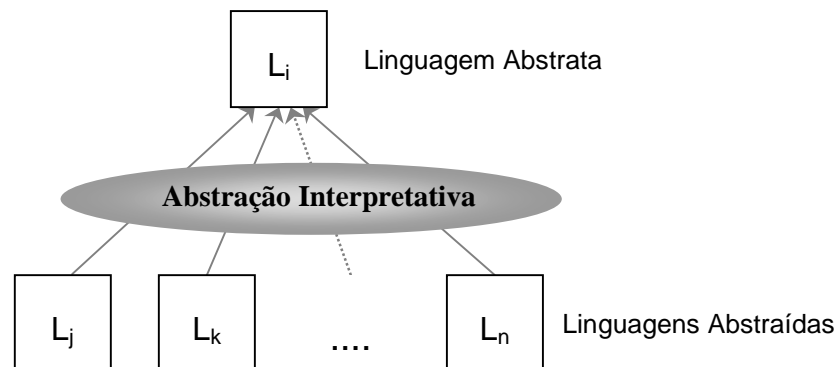


Figura 1. O Princípio da Abstração Interpretativa  
(com base em [de Souza et al.'2001])

Dado um conjunto de linguagens computacionais quaisquer  $L_i, L_j, \dots, L_n$ ,  $L_i$  é uma Abstração Interpretativa de  $L_j, \dots, L_n$  se:

- A semântica de  $L_i$  pode ser descrita pela união de todas as sentenças em  $L_j, \dots, L_n$ ; e
- Um usuário de  $L_i$  pode compreender todos os signos (lexicais e frasais) de  $L_i$  recorrendo a no máximo três recursos fundamentais e possivelmente sobrepostos:
  - (i) os padrões intrínsecos de ocorrência de tais signos em um discurso situado de  $L_i$  (sua experiência interativa ou uso da linguagem);
  - (ii) algum conhecimento metalingüístico extrínseco de  $L_i$  (a documentação do software); e
  - (iii) o seu próprio *background* (sua alfabetização computacional e seu conhecimento geral).

Aplicando-se esse princípio ao caso de aplicação extensível, podemos avaliar quando uma UIL é ou não uma abstração interpretativa de suas linguagens subjacentes, tais como a EUPL ou as linguagens de programação utilizadas pelos designers da aplicação. Podemos dizer, por exemplo, que a UIL é uma abstração interpretativa da EUPL quando sua semântica pode ser descrita completamente na EUPL e, ao mesmo tempo, um usuário consegue entender a UIL completamente sem conhecer a EUPL ou até mesmo sem saber que a EUPL existe. Ou seja, um usuário deve ser capaz de entender completamente a UIL recorrendo apenas aos padrões de signos e combinações de signos que ele encontra enquanto interage com a aplicação; às explicações disponíveis sobre a UIL, como disponibilizadas na UEL por meio de um sistema de ajuda e documentação em geral da aplicação; e à sua própria experiência com computadores, o seu conhecimento do domínio da aplicação e senso comum.

De mesma forma, para entender a EUPL um usuário não deverá precisar conhecer nada sobre as linguagens nas quais ela foi implementada, tal como as linguagens de programação usadas pelos designers da aplicação.

Ao aplicarmos o princípio da Abstração Interpretativa no sentido da EUPL para a UIL avançamos no sentido da compreensão das extensões por aqueles que não as originaram e/ou que não entendem a EUPL, como é o caso de pessoas com o perfil de trabalhador mencionado anteriormente. Considerando-se que para entender uma extensão um usuário não deverá precisar conhecer a EUPL, a aplicação desse princípio torna-se crucial no âmbito do nosso trabalho, onde as extensões devem poder ser não só construídas para uso próprio, mas comunicadas. Ou seja, esse princípio motiva a participação dos trabalhadores nos processos de extensão a uma aplicação, já que coloca ao seu alcance as comunicações relativas a esse processo. Cabe lembrar que as extensões devem poder ser compreendidas através de códigos que sejam compartilhados pelos agentes comunicantes e, como discutido na sub-seção de Linguagens de Programação por Usuários finais, a UIL se apresenta como código artificial compartilhado primário.

O princípio da Abstração Interpretativa atua no sentido do entendimento de uma extensão, uma vez que ela possua uma representação abstrata no código da UIL. No entanto, ele não garante que essa representação abstrata ocorra. Conseqüentemente, poder-se-ia construir uma

extensão que não tivesse efeito na UIL e cuja existência não seria perceptível ou compreensível aos usuários que conhecem UIL mas não atuam na EUPL. Para tratar desse problema é preciso adotar também o Princípio do Contínuo Semiótico.

### **Princípio do Contínuo Semiótico**

O princípio do Contínuo Semiótico procura avaliar as traduções de um código artificial para outro, visando um acoplamento pragmático entre eles, ou seja, observando-se que os textos traduzidos sejam válidos e pragmaticamente adequados no código de destino. O princípio avalia, por exemplo, os obstáculos para traduzir extensões criadas na EUPL em construções funcionais e usáveis da UIL.

Essas traduções terão menos obstáculos quando uma EUPL garantir a preservação de um Ciclo Mínimo de Interação, composto por três passos:

1. A aplicação diz alguma coisa para o usuário. Esse passo mostra a um usuário através da UIL estendida, por exemplo, a existência da extensão construída;
2. O usuário diz alguma coisa à aplicação. Esse passo permite que um usuário indique a ação que quer executar, como por exemplo, invocar a extensão; e
3. A aplicação responde ao usuário. Esse passo apresenta o resultado da efetivação da extensão.

A garantia desse ciclo estabelece um forte relacionamento entre a EUPL e a UIL na aplicação extensível. Tal relacionamento requer que os elementos expressos na EUPL tenham um reflexo na UIL (para que se constituam em uma extensão) e que os elementos da parte extensível da UIL (a UILx) possam ser expressos na EUPL (para que o usuário possa manipulá-los). Vejamos agora como isso se encaixa na definição formal do princípio do Contínuo Semiótico [de Souza et al.'2001, Silva'2001]:

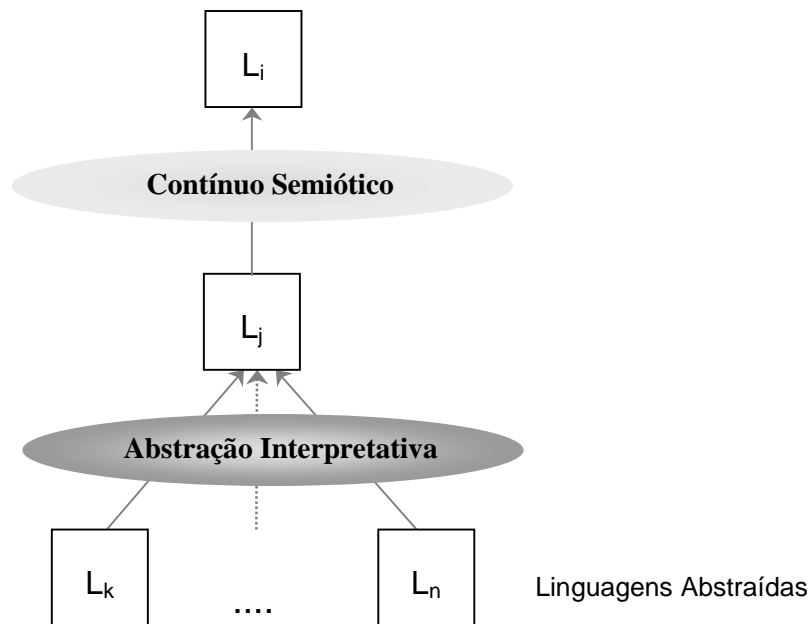


Figura 2. O Princípio do Contínuo Semiótico  
(com base em [de Souza et al.'2001])

Dadas duas linguagens computacionais  $L_i$  e  $L_j$ , elas são ditas Semioticamente Contínuas se:

- (i)  $L_i$  é uma Abstração Interpretativa de  $L_j$  – isto é,  $L_j$  provê descrições semânticas de  $L_i$ ;
- (ii)  $L_j$  é por si mesma uma Abstração Interpretativa de alguma(s) outra(s) linguagem(s) – isto é, a semântica de  $L_j$  é definida em alguma outra linguagem;
- (iii)  $L_j$  pode gerar instâncias de texto, uma organização sintática especificamente estruturada de sentenças cujo significado incorpora elementos intencionais – isto é, existe um marcador sintático para a adequação pragmática dos textos de  $L_j$  em termos de  $L_i$ ;
- (iv) Qualquer usuário que conheça  $L_i$  e  $L_j$  sempre pode traduzir uma instância arbitrária de texto em  $L_j$  em uma combinação válida de signos, realizados ou potenciais, de  $L_i$  – isto é, não existe nenhum texto pragmaticamente adequado em  $L_j$  que não possa ser formulado em  $L_i$ .

Portanto, no caso de aplicações extensíveis, a UILx é Semioticamente Contínua com a EUPL se [Silva'2001]:

1. Os usuários em geral conseguem extrair sentido dos signos da UILx e interagir com a UIL sem qualquer conhecimento da EUPL ou de sua existência;
2. Ao estender aplicações, os usuários são capazes de compreender os signos da EUPL e gerar textos na EUPL sem qualquer conhecimento de outras linguagens subjacentes a ela, tais como as linguagens convencionais de programação;
3. Existe um constituinte sintático para um texto pragmaticamente válido na EUPL; e
4. Qualquer usuário que conheça a UILx e a EUPL sempre pode traduzir uma instância arbitrária de um texto da EUPL para uma combinação válida, realizada ou potencial, de signos da UILx.

É importante notar que, para que o código da EUPL obedeça completamente o princípio do Contínuo Semiótico, é necessário que ela contenha estruturas sintáticas que garantam o acoplamento pragmático entre ela e a UIL de forma a refletir o Ciclo de Interação Mínimo anteriormente discutido.

A relação de continuidade semiótica entre as linguagens presentes em um software extensível pode ser resumida no diagrama apresentado na Figura 3.

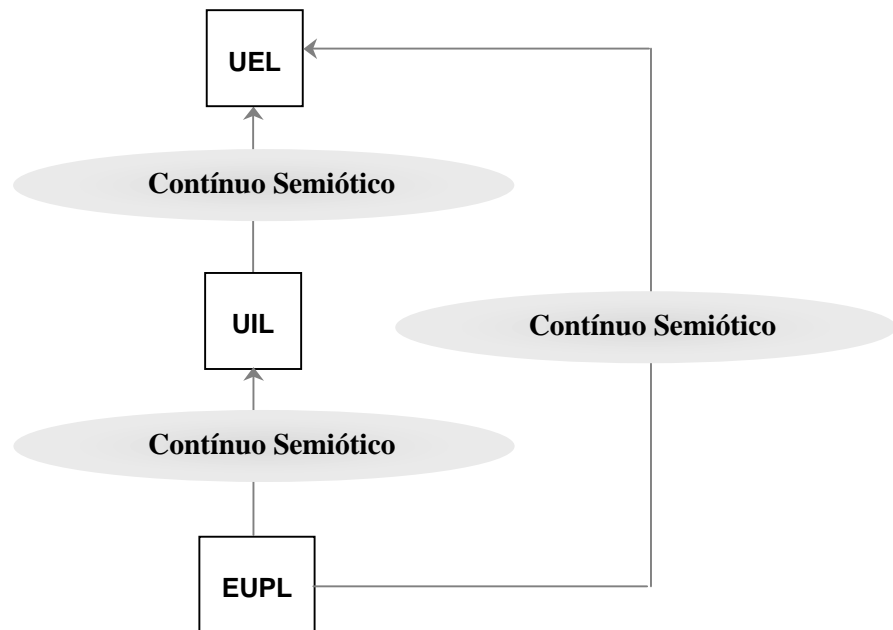


Figura 3. Continuidade Semiótica entre a UEL, a UIL e a EUPL de um Software Extensível

Observemos agora a distinção feita entre signos realizados e potenciais sob a ótica da teoria pragmática de atos de fala [Searle'1979], como feito em [de Souza et al.'2001 e Silva'2001]. A teoria considera que um discurso não só descreve e se refere a coisas, como também realiza coisas e modifica o estado do mundo. O primeiro caso é chamado de um direcionamento de adequação mundo-a-palavra e o segundo de um direcionamento de adequação palavra-a-mundo.

Quando usamos  $L_j$  (e.g. a EUPL) para descrever signos realizados na  $L_i$  (e.g. a UIL), dizemos que existirá uma relação mundo-a-palavra no direcionamento de adequação, conforme proposto pela teoria pragmática de atos de fala [Searle'1979]. Por outro lado, um signo potencial em uma linguagem é uma palavra ou frase não existente nesta linguagem que, no entanto, pode ser gerada por extensões léxicas e/ou gramaticais que obedecem aos seus padrões derivacionais — isto é, às suas meta-regras morfológicas e gramaticais. Logo, quando usamos  $L_j$  para gerar signos potenciais em  $L_i$ , estamos seguindo um direcionamento de adequação palavra-a-mundo.

Ao se comunicar sobre extensões, temos novamente a situação de um direcionamento de adequação mundo-a-palavra, com a característica de que haverá potencialmente referências a estados de mundo não só já existentes, mas estados de mundo possíveis (ou não). Logo, as mensagens sobre extensões podem ser vistas como meta-mensagens em relação às unidirecionais previstas pela Engenharia Semiótica.

A aplicação do princípio do Contínuo Semiótico, ao garantir um forte acoplamento entre a EUPL e a UIL ocasiona também uma maior eficácia nas comunicações sobre extensões, já que garante a expressão abstrata de extensões no código da UIL, que, como vimos, é o código artificial preponderante entre os agentes comunicantes.

Quando são necessárias comunicações de extensões que referenciem signos da EUPL, por exemplo durante a colaboração para construção de uma extensão, a aplicação desse princípio atua também como facilitador do aprendizado do código da EUPL, já que garante a possibilidade de desvelamento progressivo da EUPL de forma conexas à UIL, como proposto por exemplo em [DiGiano&Eisenberg'1995, Eisenberg'1995]. Da mesma forma, a aplicação do princípio pode servir como base também para mecanismos que permitam a construção de extensões a partir do código da UIL, ou seja, que se construa uma linguagem visual híbrida de programação [Nardi'1993] onde a UIL é manipulada de forma a se construírem extensões, como proposto por exemplo em [Cunha et al.'2000a, Cunha et al.'2000b].

Conclui-se então que a aplicação desses princípios é central à nossa proposta, já que maximiza a eficácia de comunicações sobre extensões, garantindo que as mesmas, depois de construídas, são expressáveis no código efetivo e preponderantemente compartilhado da UIL e são potencialmente compreensíveis pelos receptores das comunicações.

Ainda assim, cabe lembrar que extensões integram à UIL algum conteúdo potencialmente novo, inexistente na UIL original e causa da construção da extensão. A Semiótica nos ajuda a entender os processos envolvidos na criação, significação e comunicação desse novo conteúdo, através do conceito de semiótica específica.

“Uma semiótica específica é, ou tenta ser, a ‘gramática’ de um sistema de signos particular, e prova ter sucesso quando descreve um dado campo de fenômenos comunicativos como regido por um sistema de significação. Portanto há as ‘gramáticas’ da *American Sign Language*, dos sinais de trânsito, de uma ‘matriz’ de cartas de baralho para diferentes jogos ou para um jogo particular (por exemplo, pôquer). Esses sistemas podem ser estudados a partir de um ponto de vista sintático, semântico ou pragmático. Às vezes uma semiótica específica focaliza em um subsistema particular (ou s-código, como definido em [Eco’1976]) que funciona dentro de um sistema mais complexo de sistemas: como é o caso da teoria de *phonemic distinctive features* ou da descrição de oposições fonéticas pertencentes à uma dada linguagem verbal” [Eco’1984, pg. 5]. Assim podemos considerar que uma aplicação constitui uma semiótica específica de um domínio, descrevendo os fenômenos comunicativos (unidirecionais e indiretos) entre designer e usuário, e possivelmente entre usuários, de forma regida pelo código da UIL.

O conteúdo expresso por uma extensão pode atuar no escopo do sistema semiótico específico da aplicação ou representar uma evolução e/ou expansão desse sistema. Uma das forças atuantes para a construção de extensões, e conseqüentemente na comunicação sobre as mesmas, é o fato de que a interpretação de signos baseados em computador é influenciada pelo sistema semiótico circundante e que linguagens de interface integram outros sistemas semióticos existentes [Andersen’2001]. Assim, um usuário é muitas vezes capaz de perceber os limites do sistema semiótico específico da aplicação original, já que pode comparar e testar sua abrangência em relação ao(s) sistema(s) semiótico(s) circundante(s). Esse exercício de exploração dos limites de uma aplicação leva à construção de extensões, ou seja, à proposta de uma nova segmentação no plano de conteúdo e/ou de expressão.


Para a comunicação sobre a extensão, deve ser possível expressar a nova segmentação desejada e essa expressão deve estar de acordo com um código que seja compartilhado entre o emissor e o receptor da mensagem. A possibilidade e a dificuldade de expressão segundo um código disponível pode ser analisada a partir dos gêneros *Ratio Facilis* e *Ratio Difficilis* definidos em [Eco’1976, Eco’1984], que caracterizam a relação existente entre tipo e ocorrência de um signo. “Tem-se *ratio facilis* quando uma ocorrência expressiva concorda com o seu tipo expressivo, conforme foi institucionalizado por um sistema da expressão e -



como tal - previsto pelo código” [Eco’1984]. O gênero *ratio difficilis* engloba duas situações possíveis:

1. criatividade regida pelas regras - a expressão adequa-se totalmente ao código, mas muda a segmentação de conteúdos atual. É o caso em que ocorrem fatos novos inéditos. Por exemplo, se se disser pela primeira vez que “uma mulher almoçou em Vênus”, tem-se que a expressão adequa-se totalmente ao código da linguagem natural / verbal escrita e transmite um conteúdo inovador.
2. criatividade que muda as regras - o conteúdo ainda está em formação e a tentativa de expressá-lo exerce influência sobre ele. “A ausência de um tipo de conteúdo definido torna difícil elaborar um tipo expressivo; a ausência de tipo expressivo torna o conteúdo vago e inarticulado” [Eco’1976, pg. 167].

Dado que estamos trabalhando com comunicações através do canal computacional, há a possibilidade de utilizar-se mais de um código de expressão. A Semiótica, através da classificação geral de signos em símbolos, índices e ícones, informa sobre os tipos de código a disponibilizar para uma expressão eficiente de extensões.

Os símbolos são signos arbitrariamente relacionados com seus objetos e mais adaptados para exprimir correlações abstratas. Um exemplo de signo simbólico é uma palavra escrita (e.g. “casa”), já que sua relação com o conteúdo que expressa é arbitrária. Índices e ícones parecem ter uma relação mais direta com os estados do mundo, estando mais diretamente envolvidos nos atos de menção de objetos [Eco’1976]. Em termos gerais, podemos dizer que um signo icônico é culturalmente codificado, sem necessariamente implicar que seja arbitrariamente correlato ao seu conteúdo e que sua expressão seja analisável de modo discreto [Eco’1976, pg. 170]. É o caso por exemplo do ícone  que pode ser utilizado para significar o objeto telefone e cuja representação é motivada pela sua aparência física. Já um índice transmite seu significado pela indicação de causa ou efeito do mesmo. Por exemplo, a fumaça pode ser vista como signo de fogo.

A comunicação de uma extensão exige não só um esforço de expressão do emissor, mas também um esforço de compreensão dessa expressão por parte do receptor da comunicação.

Esse esforço compreende basicamente a decodificação da expressão e sua associação a um conteúdo julgado satisfatório. Essa associação passa pelo processo de Abdução [Peirce'1931, Eco'1984], no qual o receptor tenta encontrar uma regra de associação que considere adequada. Distingue-se três tipos de abdução:

- Hipercodificada - regra é dada de forma automática ou quase-automática. Esse é o caso de utilização de códigos conhecidos e estáveis.
- Hipocodificada – tem-se que selecionar dentre as regras plausíveis e já conhecidas e a seleção de uma dessas regras é feita dependendo do contexto. Por exemplo, para decidir que regra utilizar para associar conteúdo à palavra “manga”, precisamos de informações adicionais de contexto, como por exemplo “manga da camisa” ou “manga doce”.
- Criativa - a regra de associação é criada sem se saber se ela é ou não razoável. Há apenas uma intuição sobre a sua validade.

Nessa seção pudemos notar a importância que um código estável e compartilhado exerce sobre os processos comunicativos. No caso de aplicações extensíveis, isso implica no fato de que a UIL original, que é o código compartilhado, deva se manter estável apesar da construção de extensões. Cabe destacar que é através dela que o designer transmite sua mensagem unidirecional aos usuários e que essa mensagem constitui a base de conhecimento compartilhada inicial sobre a aplicação entre os diferentes agentes comunicantes de uma cultura de adaptação. Assim, corroboramos as afirmações feitas no final da seção de Programação por Usuários Finais nas quais dizemos que deve-se manter, mesmo depois de processos de extensão, ao menos uma versão da interface original que seja acessível para fins de consulta e referência pelos diferentes agentes comunicantes durante os processos de comunicação.

Esse capítulo definiu o contexto e os principais conceitos envolvidos nesse trabalho. Apresentamos agora o modelo proposto para os processos comunicativos relacionados à tarefa de extensão e depois a linguagem de representação de discursos sobre sistemas extensíveis que se baseia nos aspectos modelados.



### **3. Um Modelo Semiótico dos Processos Comunicativos Relacionados à Tarefa de EUP**

Propomos um modelo baseado em Semiótica e Engenharia Semiótica que organiza e caracteriza os processos comunicativos relacionados à tarefa de extensão. Esse modelo pode subsidiar a criação futura de métodos e ferramentas para construção de sistemas extensíveis que apoiem de forma integrada os processos comunicativos das atividades em que os usuários se envolvem ao estender software.

Como ponto de partida tomamos o trabalho de [Silva'2001], que propõe um Modelo Semiótico para Programação por Usuários Finais. O trabalho analisa as comunicações unidirecionais e indiretas que se dão a partir do designer e dos estendedores, explicadas na seção sobre Engenharia Semiótica. A análise por sua vez é baseada no Modelo de Comunicação Verbal de [Jakobson'1960].

As colocações de [Silva'2001] continuam sendo válidas aqui, dentro daquilo a que se propõem, e são complementadas em nosso modelo de forma a abarcar mensagens bidirecionais e diretas sobre extensões. A Figura 4 mostra, em fundo cinza, as complementações propostas.

### 3.1. Estrutura

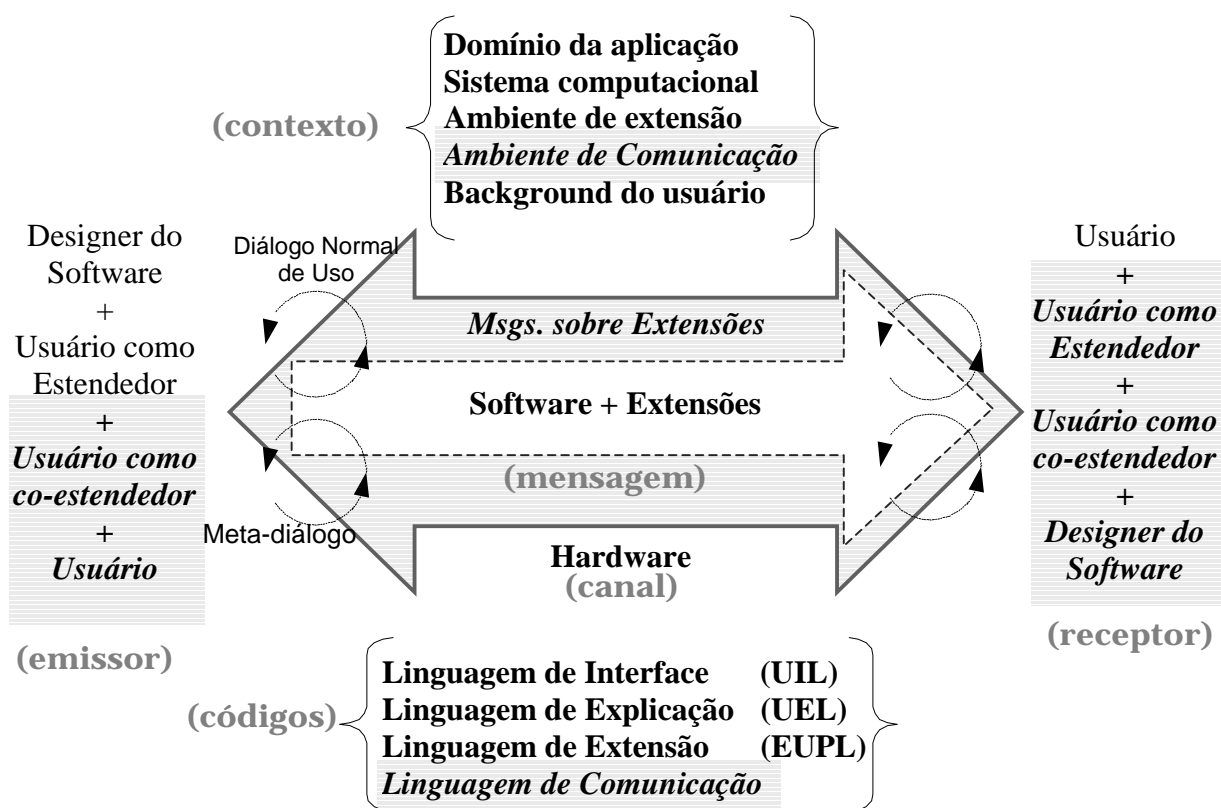


Figura 4. Extensão do Modelo Semiótico para a Tarefa de EUP

Passemos então a uma compreensão dos componentes desse modelo. Primeiramente vejamos quais são os agentes comunicantes participantes, destacando que todos eles podem atuar tanto como emissores quanto como receptores de mensagens.

#### Agentes Comunicantes: Emissor e Receptor

Os agentes comunicantes do modelo estão listados a seguir com algumas situações exemplo:

- designer de software. Além da mensagem unidirecional e de alto nível já prevista em pesquisas anteriores da Engenharia Semiótica, nosso modelo permite, por exemplo, um contato direto do designer com um estendedor para pedido de integração de uma extensão ao sistema original. O designer pode ser também o receptor de uma mensagem de um

estendedor com um pedido de ajuda para realizar a codificação de uma extensão. Em contrapartida, o designer será o emissor de uma mensagem que responderá a essa questão.

- usuário como estendedor. Na posição de emissor, um estendedor pode enviar mensagens com explicações sobre uma extensão, em conjunto ou não com a mesma, para disponibilizar o uso da extensão a outros usuários ou para responder a pedidos de ajuda de outros usuários que queiram utilizá-la. Um estendedor pode também enviar mensagens a outros estendedores em potencial, com o objetivo de uma colaboração para a construção de uma nova extensão. Um estendedor será também um receptor de mensagens durante essa colaboração.

Cabe lembrar que estamos considerando aqui diferentes tipos de colaboração, que variam de acordo com o grau de dependência existente entre os agentes comunicantes e/ou entre os objetos aos quais eles têm acesso. Pode haver o caso em que um estendedor, por um problema de segurança de acesso, se encontra impossibilitado de construir uma parte de uma extensão e portanto depende fortemente de outro co-estendedor. E pode haver também o caso em que um estendedor precisa apenas que outro lhe ajude ou ensine a realizar algo que depois ele mesmo, se quiser, poderá vir a fazer [Nardi'1993]. Nosso modelo acomoda as comunicações relacionadas a esses diferentes tipos de colaboração, embora ainda não as trate de forma diferenciada.

- usuário como co-estendedor. Esse agente comunicante é similar ao anterior, mas existente somente nos casos de construção colaborativa de extensões. Ele se distingue do estendedor pelo fato de que não foi dele a intenção inicial de construir a extensão, e sim do estendedor.
- usuário da extensão. Além da mensagem unidirecional e de alto nível já prevista em pesquisas anteriores da Engenharia Semiótica, nosso modelo permite, por exemplo, um contato direto do usuário da extensão com o criador da mesma. Como explicado anteriormente, ele pode ou não entrar em contato direto para pedir explicações sobre a extensão. Há também a possibilidade de usuários diferentes de uma mesma extensão comunicarem-se entre si para ajuda mútua. Esta possibilidade dá apoio às observações de que usuários mais experientes auxiliam os outros usuários [O'Malley'1986].

Cabe notar que nossa classificação busca distinguir os diferentes agentes de acordo com a sua função em um processo de comunicação sobre extensões. Essas funções estão mais ou menos comumente associadas às diferentes culturas mencionadas na seção sobre Práticas Colaborativas de EUP. Podemos dizer que o designer de software corresponde tipicamente à cultura do programador. Já o estendedor (principalmente) e o co-estendedor tendem a pertencer à cultura do explorador. Os usuários estão mais comumente associados à cultura do trabalhador. E o intérprete, por definição, apresenta maior flexibilidade e tende a mudar mais freqüentemente de posicionamento (estendedor, co-estendedor, etc.) em diferentes comunicações. Ao longo do tempo, os agentes podem mudar de função e de cultura, como foi observado em pesquisas de práticas colaborativas de EUP [Mackay'1990, Nardi'1993].

A análise dos agentes e dos exemplos correspondentes apontou para a relação entre o nosso modelo e os modelos de sistemas de ajuda (*help*). Vislumbramos que o modelo possa vir a ser utilizado como base de apoio para a construção de sistemas de ajuda, especialmente daqueles que se propõem a acompanhar o processo evolutivo de um software [Silveira et al.'2000]. No entanto, essa não foi a perspectiva adotada nesse estágio do trabalho. A perspectiva adotada aqui foca nos processos de criação e negociação de novos significados, cujos resultados são aplicados na construção de extensões a uma aplicação e, futuramente, poderão ser aplicados na construção e/ou evolução correspondente do sistema de ajuda dessa aplicação.

### **Canal de Comunicação**

A troca de mensagens entre esses agentes comunicantes se dá através de um canal, definido por [Jakobson'1960] como um meio físico e uma conexão psicológica entre emissor e receptor que os capacite a entrar e permanecer em comunicação. Como no modelo de [Silva'2001], temos que o canal de comunicação “será sempre o hardware, que executa o software, e seus periféricos”. Cabe ressaltar algumas propriedades do meio computacional como canal que têm impacto sobre as formas de comunicação possíveis e devem ser consideradas quando da aplicação desse modelo. São elas: a possibilidade de utilização de múltiplos códigos para a expressão de mensagens, a possibilidade de construção de mensagens interativas e a possibilidade de comunicações síncronas e assíncronas. Todas essas

características envolverão decisões de design do ambiente de comunicação e serão discutidas ao longo desse trabalho.

## Contexto

Para ser eficaz, uma mensagem requer um contexto a que se refere, apreensível pelo receptor, e que seja verbal ou suscetível de verbalização [Jakobson'1960]. Complementarmente, [Andersen'2001] explica que sistemas computacionais, tarefas e a organização do trabalho não devem ser separados. Ele nos diz que a interpretação de signos baseados em computador é influenciada pelo sistema semiótico circundante e que linguagens de interface devem ser projetadas para integrar os sistemas semióticos existentes. Além disso, ele nos diz que as linguagens de interface devem ser não só interpretáveis, mas verbalizáveis, tendo em vista que usuários se comunicam sobre suas tarefas e trabalho e sobre as tarefas e trabalho dos outros.

A partir dessas considerações, nosso modelo propõe os seguintes elementos básicos de contexto: o domínio da aplicação, o sistema computacional, o ambiente de extensão e o ambiente de comunicação. Inclui-se no ambiente de comunicação um histórico de mensagens trocadas. O objetivo do histórico é permitir que os usuários se refiram ao processo de evolução da aplicação, refletindo sobre esse processo ao mesmo tempo em que dão prosseguimento a ele.

Adicionalmente, como nota [Silva'2001], ao adotar a abordagem semiótica, o *background*<sup>1</sup> do usuário também influencia na interpretação de mensagens e portanto faz parte do contexto do receptor. Destacamos que a contrapartida do lado do emissor também é válida. Logo, o

---

<sup>1</sup> Estamos usando como definição de *background* todo o conhecimento e experiência adquiridos por uma pessoa, assim como o contexto físico e circunstancial onde ela se encontra. Essa noção está em linha com a de contexto como definida em [Brown&Yule'1983] para análise de discurso, a qual engloba o conjunto de conhecimentos do mundo necessários para a inferência de pressuposições implícitas, que influenciam não apenas a interpretação de uma mensagem, mas também sua geração.



*background* do emissor influenciará na construção de mensagens, fazendo parte de seu contexto. O (re)conhecimento dos agentes comunicantes sobre seus *backgrounds*, incluindo-se aqui o compartilhamento ou não do mesmo tempo e espaço, exerce influência sobre a comunicação.

A questão do compartilhamento de tempo e espaço tem sido abordada por estudos de sistemas de *groupware* e estudos sobre processos de comunicação. Como em [Ellis et al.'1991], apresentamos uma matriz com os tipos de interação que são derivados a partir de variações nas dimensões de tempo e espaço.

|                    | Mesmo Tempo                    | Tempos Diferentes                |
|--------------------|--------------------------------|----------------------------------|
| Mesmo Espaço       | Interação face-a-face          | Interação assíncrona             |
| Espaços Diferentes | Interação síncrona distribuída | Interação assíncrona distribuída |

Cabe ao designer a escolha de quais desses tipos de interação serão apoiados pela aplicação. Ao agente comunicante, cabe escolher, dentre os tipos de interação disponíveis, aquele que ele considere mais adequado. Essas escolhas serão influenciadas não só pela disponibilidade de recursos, mas também pelo esforço de comunicação exigido dos agentes comunicantes e por quais são os objetivos comunicativos a serem atingidos. A seguir tecemos considerações sobre esses tipos de interação, levando em conta o trabalho de [Clark&Brennan'1991] que avalia os custos relacionados aos processos de fundamentação comum entre os agentes comunicantes. Primeiramente discutiremos questões associadas aos recursos e esforços envolvidos e posteriormente avaliaremos como essas questões se relacionam com o objetivo comunicativo geral do modelo aqui proposto: o tratamento de extensões.

A Interação face-a-face é a que fornece maior quantidade de recursos para os agentes comunicantes<sup>2</sup>. Os agentes estão co-presentes, podem se ver, ver o ambiente circunstante e fazer referências a ele. Eles podem se ouvir e observar entonações, podem produzir e

---

<sup>2</sup> No escopo deste trabalho, estamos considerando agentes comunicantes ideais livres de qualquer tipo de deficiência física ou mental que possa afetar suas plenas funções comunicativas e que portanto podem se comunicar oral e gestualmente.

compreender expressões sem atraso ou até mesmo simultaneamente, e não há necessariamente interrupção entre produção e resposta de mensagens.

Por outro lado, a interação face-a-face requer a disponibilidade dos agentes para uma comunicação em um mesmo local e hora, exigindo muitas vezes um esforço antecipado de coordenação. A interação face-a-face também apresenta restrições comunicativas: ela não permite que os agentes comunicantes re-visitem mensagens trocadas anteriormente; e ela impede que um agente emissor revise sua mensagem antes de enviá-la. Por outro lado, o custo para remediar problemas posteriormente é menor nesse tipo de interação do que em outros.

Na Interação assíncrona, o local físico é mantido fixo ao longo da dimensão temporal, como é o caso de um quadro de avisos ou de troca de mensagens. O uso de referências temporais relativas, como as expressões “ontem” ou “daqui a duas horas”, exigirá um esforço adicional de codificação do contexto temporal. Nesse tipo de interação, dependendo do meio de transmissão da mensagem, os agentes podem ou não se ver e/ou ouvir. Eles podem ver e referenciar o ambiente circunstante, mas referências ao mesmo correm o risco de serem comprometidas por alterações nesse ambiente no intervalo de tempo entre a emissão e a recepção da mensagem. Naturalmente, os agentes não conseguem receber e enviar mensagens simultaneamente (e.g. ouvir uma pessoa e sorrir para ela ao mesmo tempo). Entre a produção de uma mensagem e a elaboração de sua resposta, podem acontecer uma série de interrupções (e.g. recebimento de outras mensagens), tanto no lado do emissor quanto do receptor da mensagem, ou seja, não há seqüência entre a produção e a resposta a uma mensagem. Isso causará um desvio de foco que deverá ser recuperado posteriormente para o prosseguimento da comunicação. Oportunamente, na interação assíncrona, os agentes podem rever uma mensagem mesmo depois de ela ter sido enviada, já que a mesma pode ficar disponível ao longo do tempo. O agente emissor também pode revisar sua mensagem antes de enviá-la. Tendo em vista a dificuldade e o alto custo de se remediar equívocos futuramente, esta possibilidade é útil. Finalmente, cabe lembrar que a interação assíncrona exige que os agentes comunicantes se dirijam a um local específico para enviar e receber mensagens. Conseqüentemente, para efetividade da comunicação, todos os agentes envolvidos devem ter conhecimento de qual é o local determinado e capacidade de chegar até ele.

Na Interação síncrona distribuída os agentes podem ou não se ver e/ou ouvir. No estado tecnológico atual, mesmo no caso onde os agentes podem se ver, como nas videoconferências, a visão do receptor sobre o ambiente real do emissor é mais restrita do que na comunicação face-a-face. Portanto o uso de referências ao ambiente encontra-se restringida e, quando necessário, deverá ser compensado com um esforço maior de codificação.

Devido à sincronia, normalmente não há atrasos entre a produção e a compreensão de expressões, podendo inclusive haver simultaneidade entre elas (e.g. sistemas textuais de *chat* onde ambos os agentes podem escrever ao mesmo tempo, como o comando *talk* do ambiente Unix). Normalmente há seqüência entre produção e resposta de mensagens, embora ela possa ser afetada, por exemplo, por interrupções aos agentes em cada um de seus ambientes desconexos. Há a possibilidade de os agentes reverem suas mensagens depois de elas terem sido enviadas, no entanto, não é ideal que eles façam isso durante a comunicação, já que uma demora nessa consulta pode causar espera, distração e geração de expectativas indesejadas no receptor. Da mesma forma, deve-se evitar a revisão de uma mensagem antes do seu envio. Por outro lado, comunicações ineficazes causam um custo posterior de remédio ou correção de problemas. O emissor deve procurar um equilíbrio entre essas situações.

De forma similar à interação face-a-face, a interação síncrona requer a disponibilidade dos agentes para uma comunicação ao mesmo tempo, podendo exigir um esforço antecipado de coordenação.

Na Interação assíncrona distribuída, dependendo do meio de transmissão da mensagem, os agentes podem ou não se ver e/ou ouvir. As circunstâncias físicas e temporais são desconexas e portanto qualquer referência a elas exigirá esforço extra de codificação dos agentes comunicantes. Haverá normalmente um atraso entre a produção e a compreensão de mensagens e também não há seqüência entre a produção e a resposta de mensagens. Conseqüentemente haverá um desvio de foco que posteriormente poderá ser recuperado. Tendo que os agentes podem rever uma mensagem mesmo depois de ela ter sido enviada, dada a sua disponibilidade ao longo do tempo, o agente emissor também pode revisar sua mensagem antes de enviá-la. Esta atividade é crucial, tendo em vista custo de remediação futura.

Vejamos agora a relação entre essas considerações e as comunicações que pretendemos tratar em nosso modelo. O objetivo geral das comunicações previstas é o tratamento de extensões. Deve-se levar em conta que tanto a compreensão quanto a construção de mensagens sobre extensões requerem dos agentes uma carga cognitiva considerável, especialmente pelo fato de que extensões implicam em uma modificação dos sistemas de código já conhecidos estabelecidos da UIL e da UEL. A isso acresce-se o fato de que as extensões terão um impacto sobre práticas de trabalho, logo também requerem tempo de reflexão dos agentes comunicantes. Para o atendimento dessas necessidades, é preciso que os agentes sejam capazes de rever e revisar mensagens. Ou seja, as mensagens devem persistir ao longo tempo e permitir lapsos de tempo entre o recebimento e a resposta a uma mensagem. Todos esses aspectos levam a uma primazia de comunicações assíncronas. Note-se que o histórico de mensagens é importante para a manutenção do foco e progresso na comunicação.

A escolha sobre a distribuição ou não da comunicação passa sobre os critérios de necessidade de referência ao ambiente físico e do estabelecimento de um local para troca de mensagens. Com o advento da Internet, as práticas de trabalho distribuídas são cada vez mais comuns, já que eliminam os altos custos de coordenação e deslocamento dos agentes comunicantes. Também temos que a necessidade de referência ao ambiente físico pode vir a ser compensada com um esforço de expressão do agente emissor. Assim, indicamos uma preferência geral pelas comunicações distribuídas.

Sobre o esforço adicional de codificação exigido dos agentes comunicantes, dado o não compartilhamento do mesmo tempo e espaço entre o receptor e o emissor, fazemos duas considerações. A perda de informação é inerente ao processo de codificação, onde o importante é que se mantenha um grau de informação adequado ao que se pretende comunicar. Ou seja, o emissor precisará expressar em sua mensagem os elementos que considera necessários para o entendimento da mesma, mas que não são apreensíveis pelo receptor. Caso haja falta de expressão de elementos relevantes, os receptores podem utilizar o recurso de implicatura. No entanto, quando isso acontece, as chances de falha de comunicação são maiores do que se o emissor tivesse sido explícito.

Ao realizar um esforço adicional de codificação, temos que esse esforço se converte em uma riqueza de conteúdo, que é benéfica para um uso histórico das mensagens. No caso de

processos de extensão de software, essas mensagens podem passar a servir como documentos de design das extensões e, a longo prazo, poderão ser observadas como documentos que rastreiam o processo evolutivo de um software e seu sistema semiótico específico.

No restante desse trabalho estaremos focando nessas preferências gerais, ou seja, em comunicações assíncronas e distribuídas, as quais serão utilizadas nos exemplos de aplicação do modelo. Ainda assim, todos os tipos de comunicação que possam ser realizados via computador se acomodam ao escopo do modelo.

Embora as interações face-a-face estejam fora desse escopo, não ignoramos o fato de que elas podem coexistir com o modelo. Em trabalho futuros pretendemos explorar a inclusão desse tipo de interação ao modelo, onde a parte relativa à codificação de mensagens sobre extensões poderia ser considerada por exemplo como integrante de um sistema de apoio a reuniões sobre extensões. Nesse caso, o computador não seria um canal de comunicação direta, mas um elemento de contexto e uma ferramenta de apoio.

Para concluir a descrição do componente contexto, cabe lembrar que outras informações poderiam ser atribuídas à matriz Tempo X Espaço, como detalhado em [Johansen'1991], e mais uma quantidade imensa de outras informações poderiam ser consideradas como integrantes do contexto e/ou da fundamentação comum entre receptor e emissor (e.g. tradições culturais, etc.). Tecemos aqui considerações sobre parte dessas informações que consideramos significativa e invariante entre diversos sistemas extensíveis. Assim, produzimos um modelo básico que pode ser aplicado diretamente e também estendido com outras informações contextuais.

## **Mensagem**

Para definir o componente mensagem devemos observar os conteúdos básicos a serem comunicados e suas características. Primeiramente destacamos que nosso modelo trata apenas de mensagens sobre extensões, embora não apresente critérios explícitos de separação entre essas e outras mensagens.

Os processos comunicativos relacionados à tarefa de extensão apresentam como particularidade a transmissão de unidades de conteúdo que (potencialmente) não estão presentes na aplicação original, mas que foram, estão sendo ou irão ser codificadas para integrá-la.

O conceito de semiótica específica, descrito na sub-seção 2.2, permite que se avalie a natureza do novo conteúdo. Consideramos que uma aplicação é um sistema semiótico específico de um domínio e descreve os fenômenos comunicativos (unidirecionais e indiretos) entre designer e usuário, e possivelmente entre usuários, de forma regida pelo código da UIL.

Os conteúdos de uma extensão à uma aplicação podem atuar no escopo do sistema semiótico específico da aplicação ou representar uma evolução e/ou expansão desse sistema. A evolução dos sistemas semióticos é influenciada pelo fato de que eles residem em circunvizinhança com outros sistemas semióticos [Andersen'2001]. Conseqüentemente uma aplicação tende a sofrer pressões de extensão no sentido de incorporar conteúdos não previstos em seu escopo, mas previstos no escopo de outras aplicações conhecidas pelo usuário.

Cabe destacar que o sistema semiótico específico de uma aplicação multi-usuário pode ser diferenciado para cada usuário, dependendo dos objetos e funcionalidades que são compartilhados ou não. Ou seja, diferentes usuários podem ter acesso à partes distintas de uma mesma aplicação e sua UIL. Isso caracteriza a existência de sistemas semióticos individuais distintos dentro de uma mesma aplicação. A Figura 5 representa de forma abstrata um sistema semiótico específico de uma aplicação multi-usuário e extensível. O sistema semiótico dessa aplicação é composto por três (sub)sistemas semióticos representados pelos três diferentes círculos. O sistema semiótico ao qual usuários do tipo X têm acesso é um subconjunto do sistema semiótico ao qual usuários do tipo Y têm acesso. Note-se que os usuários do tipo Y têm acesso a todo o sistema semiótico da aplicação original não estendida.

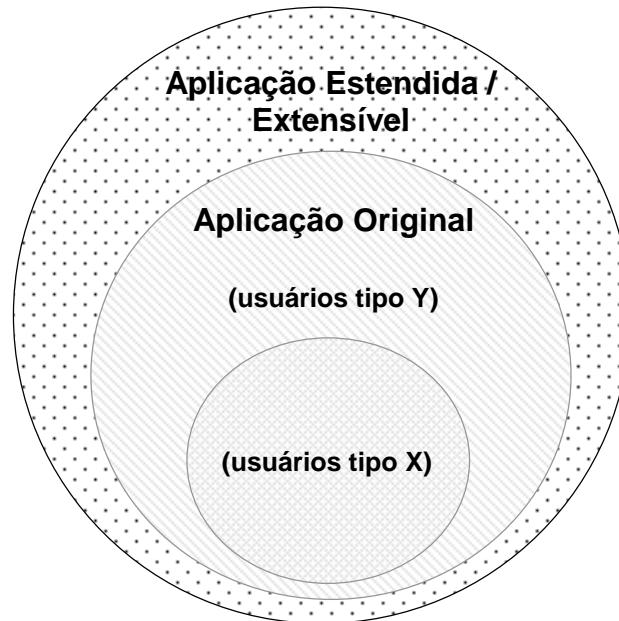


Figura 5. Sistemas Semióticos Específicos de uma Aplicação Multi-Usuário e Extensível

Imagine-se a situação em que um usuário do tipo X se aproxima da mesa de trabalho de um usuário do tipo Y e observa, na tela do computador do usuário Y, que este consegue visualizar uma informação que ele (usuário do tipo X) não vê quando usa a mesma aplicação. A observação do usuário X sobre as limitações do seu sistema semiótico em relação a um outro pode ser vista como uma motivação para que ele construa uma extensão. O objetivo dessa extensão seria incorporar, à sua parte da aplicação, a visualização que ele constatou no uso feito pelo usuário do tipo Y. Assim, poderíamos dizer que a natureza ou a origem do conteúdo dessa extensão do usuário X seria o sistema semiótico do usuário Y.

Ao longo do trabalho estaremos mencionando como sistema semiótico específico da aplicação aquele ao qual um usuário qualquer tem acesso. Quando for necessário, se destacará a diferenciação que pode existir entre os sistemas semióticos específicos de dois usuários distintos, aos quais chamaremos de sistemas semióticos específicos individuais.

Considerando-se agora também sistemas semióticos externos à uma aplicação extensível, apresentamos a Figura 6. Vemos por exemplo que o sistema semiótico específico de uma aplicação é parte integrante de um sistema semiótico computacional. Este engloba outras aplicações e softwares que podem compor o sistema semiótico computacional de um usuário

(e.g. o ambiente operacional sobre o qual a aplicação em foco é executada, editores de texto, etc.). Logo, a aplicação em foco é influenciada por essa circunvizinhança.

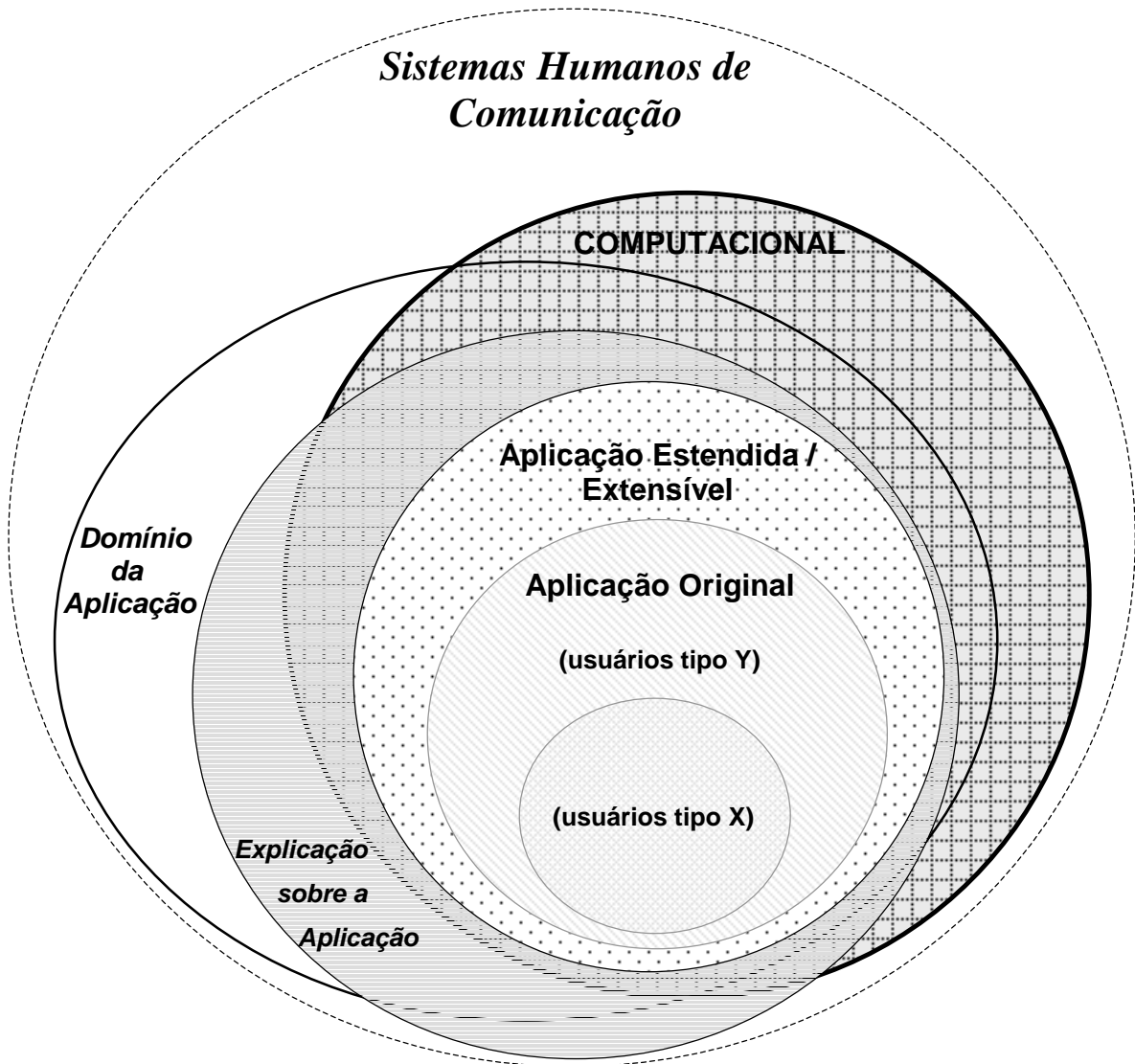


Figura 6. Sistemas Semióticos Específicos Abordados e a Semiótica da Linguagem Natural

Como explicado anteriormente, um usuário é muitas vezes capaz de perceber os limites do sistema semiótico específico da aplicação original através da comparação de suas limitações em relação ao(s) sistema(s) semiótico(s) circundante(s). O exercício de exploração dos limites de uma aplicação leva à construção de extensões, ou seja, à proposta de uma nova segmentação no plano de conteúdo e/ou de expressão. Assim uma aplicação sofre pressões



para a incorporação no seu escopo de, por exemplo, recursos e facilidades já encontrados em outros softwares e no mundo real extra-computacional que não foram previstos ou implementados na aplicação vigente. Por exemplo, veremos na avaliação empírica apresentada no capítulo 5 que uma das sugestões de extensão à uma aplicação exemplo foi a incorporação de uma funcionalidade de envio de e-mail. Essa funcionalidade não estava prevista no escopo original da aplicação exemplo, mas já existe no escopo de outras aplicações. Essa situação pressionou a construção de uma extensão que expandia o escopo da aplicação exemplo.

As diferentes naturezas de conteúdo permitem uma classificação de extensões da seguinte forma:

- extensões cujos conteúdos atuam nos limites de um sistema semiótico específico individual da aplicação, exemplificadas pela automatização de tarefas repetitivas. Em relação à Figura 6, esse seria o caso em que, por exemplo, usuários do tipo X ou Y automatizam tarefas que já são realizadas por eles rotineiramente, dentro do seu respectivo escopo de atuação. Denominamos esse tipo de conteúdo de Conteúdo Intra-Semiótico Individual,
- extensões cujos conteúdos expandem os limites do sistema semiótico específico individual da aplicação, mas não os do sistema semiótico específico da aplicação. Esse é o caso da incorporação, à um sistema semiótico específico individual, de elementos do sistema semiótico específico da aplicação (e.g. de funcionalidades existentes) que originalmente não eram apresentados na UIL daquele usuário. Em relação à Figura 6, esse seria o caso em que, por exemplo, usuários do tipo X tentariam expandir seu escopo de atuação através da incorporação de elementos já previstos no escopo dos usuário do tipo Y. Denominamos esse conteúdo de Conteúdo Intra-Semiótico Não-Individual,
- extensões cujos conteúdos expandem os limites do sistema semiótico da aplicação através da incorporação de conteúdos de outros sistemas semióticos conhecidos do estendedor, exemplificadas pela inclusão na aplicação de funcionalidades existentes

em outras aplicações do sistema computacional. Ou seja, nesse caso se dá a integração de um conteúdo de um outro sistema semiótico ao sistema semiótico da aplicação. Em relação à Figura 6, esse é o caso em que, por exemplo, usuários do tipo X ou Y tentariam expandir seu escopo de atuação através da incorporação de elementos do sistema semiótico computacional ou do mundo real (e.g. linguagem natural). Denominamos esse tipo de conteúdo de Conteúdo Extra-Semiótico Segmentado, e

- extensões que expandem os limites do sistema semiótico da aplicação de forma criativa e não relacionada a elementos de outros sistemas semióticos conhecidos do estendedor. Esse é o caso em que, por exemplo, usuários do tipo X ou Y tentariam expandir seu escopo de atuação através da incorporação de elementos por ele criados não existente ainda em nenhum sistema semiótico. Denominamos esse tipo de conteúdo de Conteúdo Extra-Semiótico Não-Segmentado.

A aplicação do nosso modelo em instâncias de domínios e grupos de usuários implica na especialização da representação mostrada na Figura 6 com o detalhamento dos sistemas semióticos pertinentes. Veremos um exemplo de como isso pode ser feito no capítulo 5 quando se faz uma avaliação empírica do trabalho.

Para a comunicação da extensão a outros agentes comunicantes, deve ser possível expressar o seu conteúdo e também a sua intenção de design (e.g. para que ela serve), além da intenção de comunicação (e.g. pedir uma ajuda/colaboração) utilizando-se um ou mais códigos que sejam compartilhados entre o emissor e o receptor da mensagem.

### **Códigos de Comunicação**

Tendo em vista que o canal computacional é uma mídia que admite múltiplos códigos, caberá ao designer da aplicação escolher que códigos serão disponibilizados aos agentes comunicantes. Como vimos na seção 2.2, a Teoria da Produção Sínica [Eco'1976] nos informa sobre diversos tipos de signos utilizados nos códigos para comunicação, mostrando que diferentes tipos de signos são mais ou menos adequados para a transmissão de

determinados tipos de conteúdo. Ela fala também dos custos associados à produção dos signos. Essas informações podem servir como guia para os designers durante decisões sobre os códigos a serem disponibilizados. E podem ser úteis também para os agentes comunicantes tanto na escolha de quais códigos (dentre os disponíveis) utilizar, quanto no planejamento de forma geral dos custos envolvidos na comunicação. Posteriormente, pode-se pensar em utilizar essa informação para avaliações sobre o uso dos códigos nos ambientes de comunicação sobre extensões.

Primeiramente temos que a escolha de um repertório de tipos de signos preestabelecidos por um código conhecido, a fim de produzir uma ocorrência de um tipo preciso, é menos custosa do que a invenção de um novo tipo de signo. Portanto, temos que é preciso que um designer disponibilize códigos para serem utilizados pelos agentes comunicantes de forma a evitar a necessidade de os agentes criarem novos códigos (como é o caso por exemplo do desenho de figuras).

Analisando separadamente a parte da mensagem que corresponde à intenção de comunicação e de design da extensão, observamos que as mesmas são idéias abstratas e portanto tendem a ser preferencialmente expressas a partir de códigos simbólicos. A linguagem natural é um sistema semiótico simbólico altamente expressivo, compartilhado pelos agentes comunicantes. Portanto, a mesma pode ser utilizada para expressar as intenções. A linguagem natural permite um alto grau de expressividade, mas também a ocorrência de ambigüidades. Ela também não é totalmente processável pelo computador, fator que restringe a possibilidade de processamento de informações nela codificadas.

Poderia se pensar na criação de uma linguagem formal para expressão da intenção de design da extensão, de forma a diminuir ambigüidades e permitir processamentos computacionais sobre as mesmas. No entanto, para isso seria necessário um estudo mais aprofundado sobre padrões de extensões, de forma análoga ao trabalho de Searle, entre outros, nos quais se mapeou um conjunto de intenções de comunicação. Portanto, nas aplicações correntes do modelo, utilizamos a própria linguagem natural para expressar uma intenção de design de extensão.

Para se comunicar sobre extensões, uma representação das mesmas é imprescindível. Como explicado anteriormente, as extensões são objetos que têm relação direta com os estados do mundo (o domínio e a aplicação) e portanto tenderiam a ser expressas através de ícones ou índices. No entanto, outro aspecto importante recai sobre a representação das extensões: o fato de que ela deve ser ao mesmo tempo compreensível entre os agentes comunicantes e processável pelo computador.

Cabe notar que o (multi)código da UIL já é utilizado para referir aos estados do mundo do domínio da aplicação, ao mesmo tempo em que é compartilhado entre os agentes comunicantes e é processável pelo computador. Tendo que é preferencial a escolha de signos conhecidos à criação ou aprendizado de novos, é interessante que, sempre que possível, a extensão seja expressa em termos do código da UIL. Assim, minimiza-se o esforço do emissor à codificação e ao destaque do novo significado que está sendo introduzido pela extensão e que não faz parte da UIL original. A contrapartida do lado do receptor quanto à decodificação e compreensão da mensagem também é válida.

Como veremos na próxima seção, dependendo de qual é a relação existente entre o novo significado e as formas de expressão previstas na UIL, a comunicação pode ser mais ou menos custosa. Para analisar isso, discutiremos mais profundamente as características dos códigos sob uma perspectiva de utilização dos mesmos.

### **3.2. Dinâmica**

A explicação do modelo e de seus componentes mostrou quais são as forças atuantes nos processos de comunicação sobre extensões; no entanto, ainda falta descrever o comportamento dinâmico temporal dessas forças.

Primeiramente definimos o escopo de análise da dinâmica dessas forças nesse trabalho. A Figura 8 organiza um processo de colaboração em uma espiral onde inicialmente se negociam os significados. Quando entende-se que essa negociação foi satisfatória (ou seja, ambos emissor e receptor estão convencidos de que têm interpretantes consistentes sobre um conteúdo específico), então pode-se dizer que houve comunicação. A partir desse ponto, o foco se volta para a negociação de uma ação a ser tomada. A comunicação passa a ocorrer em

um nível onde os entendimentos dos significados são tomados como pressupostos. Quando, durante a negociação da ação, percebe-se um problema de convergência de interpretante, inicia-se uma nova espiral de comunicação. Logo, alternam-se ciclos de negociação de significados ou comunicação, com ciclos de negociação de ação ou tomada de decisão até o fim da colaboração.

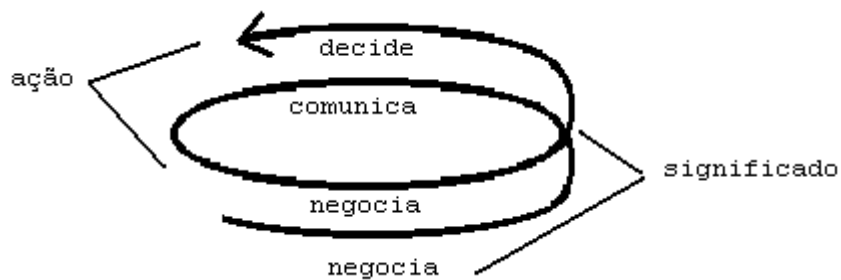


Figura 8. Espiral de Colaboração: Comunicação e Tomada de Decisão

Esse trabalho se concentra no primeiro ciclo dessa espiral, onde se dá a negociação e comunicação de significados. Para o outro ciclo, apontamos para as pesquisas existentes de sistemas de apoio à negociação e tomada de decisão [Winograd'1987, Conklin&Begeman'1988, Laufer&Fuks'1995]. Dentro do nosso escopo, subdividimos o fenômeno de comunicação em três passos principais: a elaboração de uma mensagem, a sua transmissão e a sua compreensão, e analisamos o primeiro e o terceiro passos.

### **Elaboração de Mensagem**

A elaboração de uma mensagem é o processo no qual se dá o mapeamento de um conteúdo para um código expressivo que seja aceitável e compreensível tanto pelo emissor quanto pelo receptor da mensagem.

Na descrição do elemento código do modelo discutimos a expressão de intenções em mensagens sobre extensões. Agora iremos analisar a expressão das extensões propriamente ditas, adotando as visões sintática, semântica (lexical e do modelo da aplicação) e pragmática que pode-se ter de um sistema semiótico específico, já que esse é regido por um código.

Consideramos como ponto de partida o código da UIL original que é acessado pelo emissor da mensagem. Vejamos então em que visão a expressão de uma extensão pode atuar:

- a extensão é totalmente expressa a partir da criação de novo(s) elemento(s) do léxico da UIL. Denominaremos esse tipo de expressão de Expressão Atuante no Léxico,
- a extensão é totalmente expressa a partir da criação de nova(s) estrutura(s) sintática(s) e/ou elemento(s) do léxico da UIL. Denominaremos esse tipo de expressão de Expressão Atuante na Sintaxe,
- a extensão é totalmente expressa da criação de novo(s) elemento(s) semântico(s) e/ou estrutura(s) sintática(s) e/ou elemento(s) do léxico da UIL. Denominaremos esse tipo de expressão de Expressão Atuante na Semântica, e
- a extensão não é totalmente expressável, pois seu conteúdo ainda está em formação e a tentativa de expressá-lo exerce influência sobre ele. Denominaremos esse tipo de processo de expressão de Expressão Atuante na Pragmática.

No capítulo de Avaliação exploraremos essas distinções para classificar as extensões que foram comunicadas por usuários finais em uma primeira investigação empírica.

Para atingir esses diferentes tipos de expressão o usuário precisará utilizar um código que atue nessas diferentes visões da UIL: a EUPL. Diferentes propostas de EUPL dependem de diversos critérios de decisões de design, dentre os quais destacamos: o custo de aprendizado do usuário e a expressividade da linguagem.

Uma EUPL que permite a expressão de extensões a partir da UIL (como por exemplo as linguagens de gravação de macro) possui um baixo custo de aprendizado, porém, apresenta também uma baixa expressividade. Já EUPLs com expressividade equivalente à máquina de Turing normalmente apresentam um alto custo de aprendizado. Em ambos os extremos podem ocorrer situações onde um usuário não consegue expressar uma extensão. No primeiro caso, a baixa expressividade o impede. No segundo caso, o alto custo de aprendizado o leva a desistência.

Falemos portanto sobre os casos onde a comunicação é motivada pela impossibilidade ou incapacidade de um estendedor expressar uma extensão totalmente, por si só, e em um código processável por computador. Para esses casos, é preciso prover códigos adicionais para que os estendedores tenham uma forma alternativa de especificar a extensão e se fazer entender pelo receptor de sua mensagem. Ainda assim, é possível que o estendedor seja capaz de expressar parte da extensão a partir do código da UIL ou da EUPL. Logo, dividiremos essas comunicações em dois tipos: extensões parcialmente construídas e extensões não construídas. (Note-se que, ao incluir-se expressões da EUPL nas mensagens, pressupõe-se que o receptor seja capaz de compreendê-las.)

Nas extensões parcialmente construídas os estendedores devem ser capazes de combinar expressões do código da UIL e da EUPL com expressões de um outro código, em um processo de extra-codificação. O ideal é que esse código tenha um baixo custo de aprendizado (como por exemplo a LN), dado o custo de aprendizado que uma aplicação extensível por si só já exige. É importante também que esse código permita fazer referência às expressões da UIL e/ou da EUPL, permitindo assim ao emissor uma contextualização e controle de foco sobre a comunicação e a colaboração proposta. Um exemplo seria a construção de uma mensagem expressa através de um texto da UIL (uma interface ou seqüência de passos de interação) combinado com anotações em linguagem natural. Na resposta a essa mensagem, as anotações seriam re-escritas em códigos processáveis por computador (e.g. a EUPL) e a construção da extensão seria finalizada.

Para os casos de extensões não construídas, nem mesmo parcialmente, além da intenção de design da extensão, pode ser comunicada também algum tipo de explicação mais detalhada sobre ela, lembrando sempre da utilização de um código que seja compreensível pelo receptor da mensagem, potencialmente a LN.

Ao se comunicarem, emissores utilizam diferentes funções da linguagem. [Jakobson'1960] mostra que cada um dos componentes do modelo de comunicação determina uma função diferente da linguagem. Essas funções são utilizadas em conjunto em uma mensagem e a predominância de uma função em relação a outra ocorre de acordo com o tipo de informação que o emissor pretende transmitir ou obter.

Primeiramente detalhemos função referencial (ou denotativa). Ela enfoca o contexto da comunicação, informando sobre o objeto do discurso. Como nota [Jakobson'1960], o emprego dessa função predomina em numerosas mensagens. No âmbito do nosso trabalho, o enfoque das comunicações será primordialmente a extensão, ou seja o objeto do discurso. Portanto, é imprescindível que se preveja e apóie o uso dessa função por parte dos agentes comunicantes.

Em nosso trabalho, o contexto de atuação direta dos agentes comunicantes é o ambiente de comunicação. O objeto comum aos discursos tratados é a extensão do software. Assim, devem estar disponíveis no contexto do ambiente de comunicação recursos que informem sobre a extensão, tais como referências diretas à mesma ou a partes dela e também a intenção de design da extensão. Esses recursos permitirão a construção do discurso e a manipulação de foco ao longo do mesmo. Como consequência, o ambiente de comunicação deve prover representações das extensões que possam ser integradas ao discurso em elaboração. Em contrapartida, essas representações e as referências a elas devem ser perceptíveis e compreensíveis pelo receptor da mensagem.

No âmbito da comunicação entre usuário e software, o trabalho de [Silva'2001] descreve a importância da utilização da função referencial para chamar a atenção dos objetos sobre os quais os agentes estão atuando em determinado momento e em que contexto. Como parte integrante do software, o mesmo vale para o ambiente de comunicação aqui proposto. Primeiro, deve-se diferenciar o mesmo dos demais ambientes do software. Adicionalmente, dentro do próprio ambiente de comunicação, é preciso distinguir os diferentes contextos, ou modos de operação, possíveis: o de leitura de uma mensagem, o de elaboração de uma mensagem inicial e o de elaboração de uma mensagem de resposta. Em cada um desses modos, os objetos disponíveis e a forma de atuação sobre eles pode mudar. Portanto, o ambiente deve deixar claro para o usuário qual é o modo de operação vigente a cada momento.

A função expressiva (ou emotiva) enfoca o emissor da mensagem. Ela visa a expressão direta da atitude de quem fala em relação àquilo que está falando. No ambiente proposto, os agentes comunicantes devem poder manifestar suas posições em relação às extensões. Por exemplo, criadores de extensões podem se mostrar entusiasmados ao anunciá-las ou, durante um



processo de construção colaborativa, podem querer argumentar contra ou a favor de soluções propostas.

A função conativa (ou apelativa) enfoca o destinatário. Ela evoca uma ação do receptor da mensagem e, normalmente, está relacionada ao vocativo e ao imperativo na linguagem verbal. No processo de comunicação sobre extensões, podemos relacionar o uso dessa função a pedidos de colaboração. Esses pedidos podem ter diferentes motivações, como por exemplo um pedido de construção de extensão em parceria, ou até mesmo um pedido de explicação sobre uma extensão já construída. Nesse último caso pode haver um emprego conjunto da função conativa com a função metalingüística.

A função metalingüística enfoca o código empregado na comunicação. Como dito em [Silva'2001], ela está direcionada à necessidade do emissor de verificar se ele e o receptor estão usando o mesmo código e será empregada na elucidação de problemas de interpretação do código. Como discutido anteriormente, nosso modelo, através da bidirecionalidade de comunicação, complementa o de [Silva'2001] já que não há, *a priori*, uma limitação ao uso da função metalingüística. No entanto, o uso dessa função não deve ser feito de forma indiscriminada, dadas as desvantagens associadas ao mesmo, entre as quais podemos citar: o desvio do foco do discurso do tema central -as extensões- para o código e o custo associado a elaboração de mensagens adicionais e recuperação do foco anterior. Para minimizar o uso da função metalingüística, é preciso apoiar os agentes comunicantes na elaboração de mensagens, de forma que seja utilizado, sempre que possível, um código que seja compartilhado entre o emissor e o receptor da mensagem, e que também estejam disponíveis recursos de explicação sobre a linguagem, para o caso de quando o receptor não compreender o código.

A função fática enfoca o canal de comunicação, com o propósito de testá-lo. É o caso, por exemplo, de quando uma pessoa pergunta à outra pelo telefone: *Você consegue me escutar?* No âmbito desse trabalho, fala-se da troca de mensagens assíncronas via computador. A função fática pode ser empregada, por exemplo, para saber se um destinatário recebeu ou não uma mensagem já enviada. Assim, requer-se do ambiente de comunicação a disponibilização de recursos que permitam o emprego da função fática. Um exemplo de recurso seria o uso de uma confirmação de recebimento de mensagem por parte do receptor. Cabe lembrar que

algumas comunidades ou indivíduos se mostram resistentes à aplicação de determinados recursos deste tipo. Portanto, cabe ao ambiente disponibilizar esses recursos mas, ao mesmo tempo, deixar a cargo da comunidade de usuários uma decisão sobre quais recursos utilizar e de que forma.

A função poética enfoca a própria mensagem, valorizando sua forma de expressão. No ambiente que estamos propondo há o objetivo de que as comunicações se dêem de forma eficaz, com uso mínimo da função metalingüística. Para isso, é necessário incentivar o emissor da mensagem a priorizar formas de expressão que sejam compreensíveis pelo receptor da mensagem, tais como as que já são utilizadas no próprio software. Este é portanto um requisito para o ambiente de comunicação.

A análise das funções nos mostra requisitos que devem ser atendidos por um ambiente de comunicação sobre extensões e pelos códigos disponíveis no mesmo. Ao se utilizarem códigos que sejam compartilhados entre os agentes comunicantes, minimiza-se o uso da função metalingüística e aumenta-se as chances de eficácia na comunicação.

A análise e discussão acima elucidou aspectos da elaboração da mensagem, que obviamente estão relacionados com aspectos da compreensão da mesma, depois que ela é enviada. A seguir detalhamos os aspectos envolvidos no passo de compreensão de uma mensagem.

### **Compreensão de Mensagem**

O primeiro passo para compreensão de uma mensagem é a sua decodificação, ou seja, a leitura de um texto e a derivação de seu(s) interpretante(s). Para entendermos essa passo, precisamos aplicar o conceito de abdução. Suponhamos que o receptor “leia” somente uma extensão já pronta. Para as partes da extensão codificadas em contínuo semiótico com a UIL original será feita uma abdução do tipo hipercodificada, ou seja, o mapeamento entre expressão e significado será automático ou quase automático. Para as partes da extensão que estendem o código da UIL, será aplicada a abdução criativa, onde o receptor terá uma intuição sobre a regra de mapeamento, mesmo sem saber se ela é razoável. Se estiverem incluídas ou forem adicionadas à extensão informações explicativas em linguagem natural, então, a partir

dos textos em LN o receptor poderá gerar uma ou mais regras plausíveis para o mapeamento e, ao considerá-las, ele estará em um processo de abdução hipocodificada.

Os custos de compreensão aumentam na medida em que se passa da abdução hipercodificada para a hipocodificada e para a criativa. Caso o receptor não alcance uma compreensão, então ele poderá iniciar a elaboração de uma mensagem para o emissor na qual ele fará uso da função metalingüística de forma a negociar significados e tentar encontrar a regra de mapeamento. Nessa mensagem ele pode ou não explicitar as regras hipotéticas que foram levantadas no processo de abdução.

Quando o receptor (se convence de que) compreende a mensagem recebida, ele está apto a passar para o passo seguinte, que pode ser uma tomada de ação e ou decisão seguida ou não da elaboração de uma mensagem de resposta.

Assim concluímos essa seção e esse capítulo. Nele apresentamos um modelo dos processos de comunicação sobre extensão baseado nas teorias da Semiótica e da Engenharia Semiótica. Mostramos como estendemos o modelo descrito por [Silva'2001] de forma a abarcar, através do mesmo canal computacional, as comunicações bidirecionais e diretas. Por conseguinte, reavaliamos os aspectos estruturais e funcionais do modelo.

Estruturalmente, percebemos uma variação maior no conjunto de agentes emissores e receptores e nos possíveis contextos de comunicação entre esses agentes. Discutimos como as noções de tempo e espaço atuam sobre o contexto e indicamos uma preferência por comunicações distribuídas e assíncronas, dada a vantagem de minimizar-se esforços de coordenação e deslocamento dos agentes e, principalmente, dada a necessidade de revisão e reflexão das mensagens trocadas. Essa necessidade advém de uma característica básica da extensão como objeto de discurso: o fato de que ela introduz um novo conteúdo ao discurso preconizado pela mensagem inicial do designer, destacando-se que esta corresponde à (parte da) base de conhecimento compartilhada entre os agentes comunicantes e que portanto não deve ser destruída. Apresentamos uma classificação de extensões de acordo com a natureza de seu conteúdo e depois distinguimos formas de expressão de um conteúdo tomando como base

o código da UIL. Finalmente descrevemos os passos de elaboração e compreensão de mensagens que compõem os aspectos dinâmicos do modelo.

A seguir propomos uma linguagem artificial projetada com base no modelo. Mostramos que a partir dela, aplicando-se os conceitos de Abstração Interpretativa e Contínuo Semiótico, pode-se derivar os códigos a serem utilizados e compartilhados entre os agentes comunicantes. O referencial comum entre códigos pode maximizar a compreensão dos mesmos e conseqüentemente minimizar o uso da função metalingüística, avançando-se na busca por uma eficácia nos processos de comunicação.

## 4. Linguagem de Representação de Discursos Baseada no Modelo (X-DIS)

Nossa linguagem atua como um elemento integrador, sendo a base referencial comum entre os códigos dos diferentes ambientes do software: o de uso normal, o de uso extensível e o de comunicação. Podemos dizer que o código de uso normal do software – a *User Interface Language* (UIL) – é o código potencialmente compartilhado entre os agentes comunicantes. A linguagem, atuando como base comum, permite que os agentes comunicantes re-aproveitem os conhecimentos adquiridos em um código (e.g. o da UIL) para o aprendizado dos outros (e.g. os códigos dos ambientes de extensão e comunicação).

Cabe lembrar que os códigos permitirão aos usuários reconhecer e atuar sobre os elementos do domínio da aplicação para tentar expressar e/ou atingir os objetivos desejados. Assim, como base para esses códigos, a representação precisará fazer referência a elementos do domínio.

Diferentes domínios são passíveis de diferentes formas de representação que podem se mostrar mais ou menos adequadas. Por isso, é importante caracterizar os domínios tratáveis pela representação que propomos. Para isso, definimos o conceito de Sistemas Orientados a Tarefas: são aqueles que implementam planos de como se atingir determinados objetivos através de tarefas que podem ser sub-divididas em uma rede de passos de interação usuário-sistema [Cunha et al.'2000a, Cunha et al.'2000b]. Neste trabalho estaremos tratando de domínios que possam ser formulados de acordo com esse conceito.

Sistemas Orientados a Tarefas são interessantes para esse trabalho no sentido em que se pode considerar que é inerente aos mesmos a necessidade de construção de extensões, dada a distância existente entre planos e ações situadas [Suchman'1987]. Em relação às comunicações sobre extensões, destacamos um caso especial de Sistemas Orientados a Tarefas: os sistemas de *workflow*. Em sistemas de *workflow* há também uma definição, para cada tarefa, de quem ou que será responsável pela realização da mesma. Por isso, em alguns

casos de extensão, a realização de comunicações bidirecionais será não só desejável, mas imprescindível. Assim, no restante do trabalho, exemplificaremos como a representação apóia sistemas de workflow, lembrando que a mesma pode ser utilizada para Sistemas Orientados a Tarefas em geral, ao relaxar-se a definição de responsáveis pelas tarefas.

Façamos então uma definição de sistemas de workflow: são sistemas que implementam fluxos de trabalho que visam atingir objetivos de negócio. Mais precisamente, segundo a *Workflow Management Coalition* (WfMC), sistemas de workflow são automações de processos de negócio nas quais documentos, informação e tarefas são passadas de um participante para outro de acordo com um conjunto definido de regras para alcançar ou contribuir para um objetivo de negócio [WfMC'1995, WfMC'1999]. Esses sistemas podem ser vistos como a fixação em ambiente computacional dos objetivos de negócio a serem atingidos e dos planos previstos para a realização dos mesmos dentro de contextos antecipados. Algumas características destes planos são:

- eles podem ser representados como uma rede interconectada de passos a serem seguidos para a realização de um objetivo,
- cada passo pode ter um conjunto de pré e pós-condições para a sua execução,
- a realização de cada passo pode envolver o processamento de estruturas de informação do domínio, e
- cada passo possui um conjunto de papéis (grupos de pessoas caracterizadas e sistemas automatizados) capazes de executá-lo, onde diferentes passos de uma mesma rede podem envolver diferentes papéis.

Como representação formal e computável desses planos, propomos a utilização de *Augmented Transition Networks* (ATNs) [Woods'1970], devido à correspondência entre elas e as três primeiras características listadas acima. Note-se que a definição de um conjunto de papéis capaz de executar um passo pode ser transformada em uma pré-condição para realização do mesmo. Assim, a representação aqui proposta constitui uma linguagem que prescreve e também processa planos.

A execução de planos envolve processamento de estruturas do domínio. Descreveremos brevemente os modelos que representam essas estruturas e são integrados na componente semântica da linguagem formal. O objetivo dessa descrição não é detalhar extensivamente os modelos, dado que não constituem o foco desse trabalho, mas prover uma base para o entendimento da linguagem formal e dos elementos que fazem parte dos discursos dos agentes comunicantes. Basicamente temos o Modelo de Domínio e o Modelo de Interface.

Todos os modelos adotados seguem, e em alguns casos estendem, propostas já descritas em outros trabalhos. Sempre que possível utilizamos modelos que, assim como a nossa proposta, se enquadram dentro do Framework da Engenharia Semiótica. Dessa forma motivamos uma maior consistência entre os modelos e a linguagem formal, dado o compartilhamento da visão de que o software é uma meta-mensagem enviada de seu designer para os usuários finais. Quando isso não foi possível, de forma a facilitar o entendimento do leitor, optamos por modelos com reconhecida aceitação na indústria de software.

A Figura 9 a seguir apresenta um esquema geral para as explicações que se seguem.

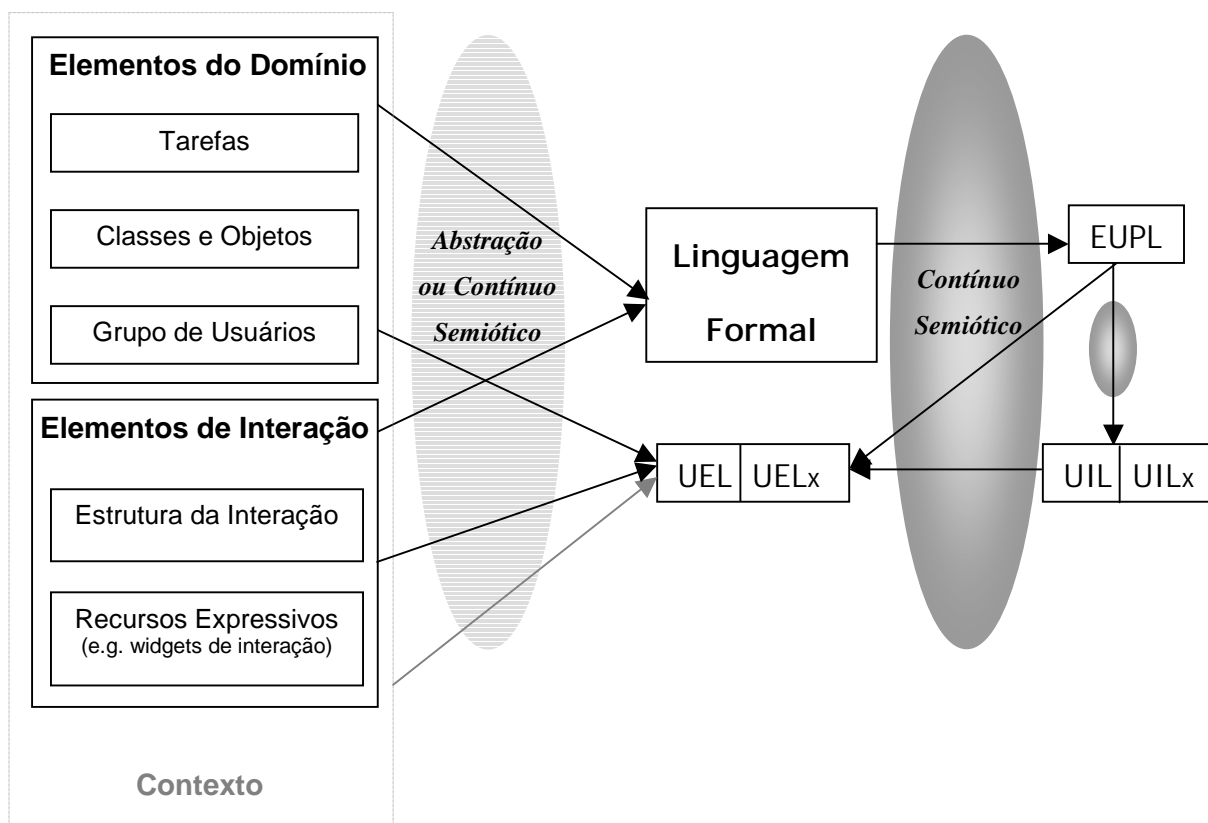


Figura 9. Elementos de Discurso, A Linguagem Formal e Linguagens Derivadas

Para exemplificar a utilização dos modelos e da linguagem formal selecionamos o domínio de Empréstimos de Materiais, cuja implementação foi realizada em ambiente WWW com recursos de processamento tanto na máquina cliente como na máquina servidora.

#### 4.1. Modelo de Domínio

O modelo de domínio foi subdividido em: Modelo de Tarefas e Modelo de Informação. Para o caso de domínios de trabalho colaborativo, como por exemplo os sistemas de workflow, se faz necessária também uma representação de um Modelo de Grupo.



## Modelo de Tarefas

O Modelo de Tarefas apresentado aqui seguiu a proposta da WfMC [WfMC'1995] e a estendeu de forma a explicitar também a relação entre tarefas e os passos de interação usuário-sistema que as realizam.

O Modelo de Tarefas é uma representação *top-down* de um domínio onde, no nível mais abstrato, descreve-se quais são os Processos de Negócio de uma organização. Estes processos podem ou não ser divididos em outros (sub)processos. A Figura 10 mostra, de forma abstrata, as principais classes e relações que compõem o Modelo de Tarefas. A notação utilizada é a proposta pela *Unified Modeling Language* (UML) [Booch et al.' 1999]. Podemos ver que cada processo (ou sub-processo) é detalhado por uma ou mais atividades (notação 1..\*). Cada atividade pode ser seguida por zero ou mais atividades (notação \*), formando-se assim uma Rede de Atividades.

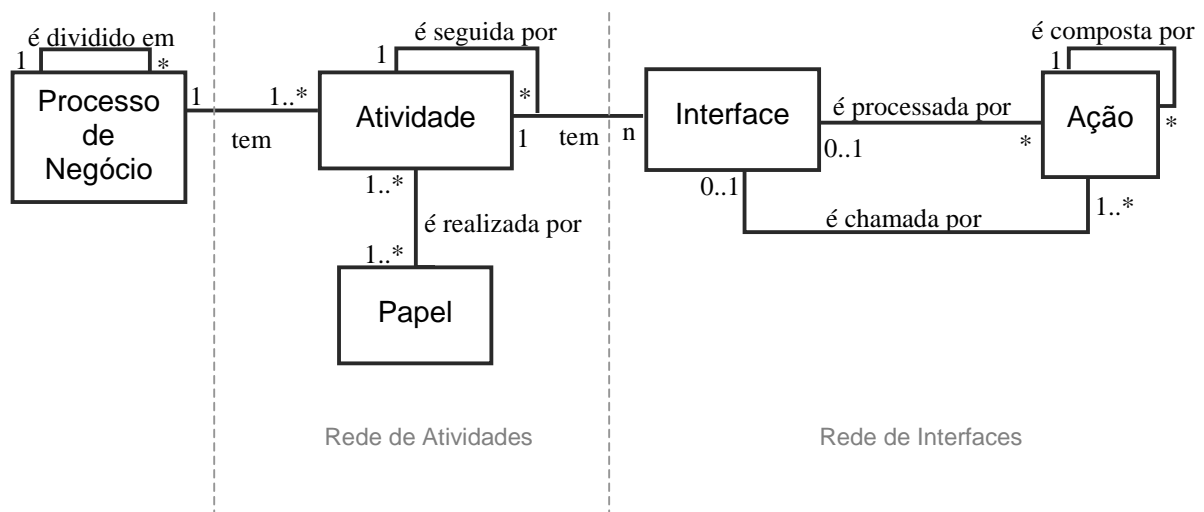


Figura 10. Mapa do Modelo de Tarefas

Cada atividade representa um passo lógico dentro de um processo e possui um conjunto definido de um ou mais papéis responsáveis por realizá-la. Cada papel pode ser associado a um conjunto de pessoas, normalmente usuários do sistema de workflow. Assim, pode-se

variar o conjunto de pessoas responsáveis por atividades com baixo impacto na representação e funcionamento do sistema.

A Figura 11 mostra as atividades do domínio de Empréstimo de Materiais, que será usado como exemplo nesse trabalho. Cada retângulo representa uma atividade, mostrando o seu nome e, abaixo da linha divisória, o conjunto dos papéis que a realizam. Nesse domínio nenhuma atividade era seguida de outra, portanto não houve necessidade de representar relações de seqüência entre elas. Caso essas relações existissem, elas seriam expressas através de linhas conectando as atividades. Está fora do escopo do modelo da WfMC uma representação explícita de outras relações possíveis entre atividades.

#### Rede de Atividades

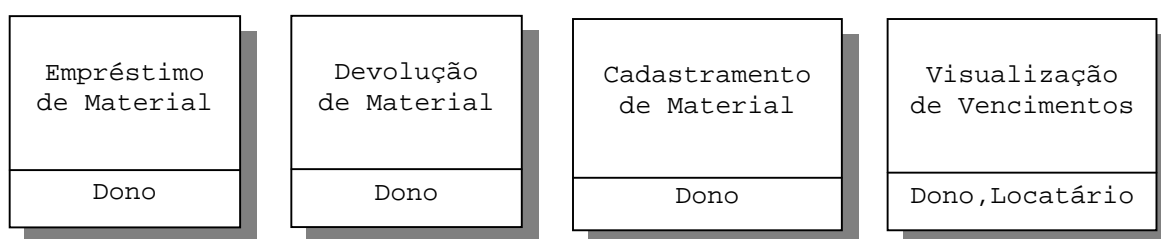


Figura 11. Rede de Atividades do domínio de Empréstimo de Materiais

Também está fora do escopo do modelo da WfMC uma representação dos passos de interação usuário-sistema que levam à realização de atividades. No entanto, esse aspecto é essencial dentro de nossa proposta, tendo em vista nosso foco na comunicação através da utilização de códigos compartilhados e na manipulação e extensão dos códigos disponíveis como forma de manipulação e extensão de conteúdos do domínio da aplicação. Para representar a relação entre cada atividade e a sua realização no nível de interface, adicionamos mais um nível de detalhamento ao modelo da WfMC. Voltando à Figura 10 vemos que cada atividade é transformada em um conjunto de interfaces. Cada interface, após comando do usuário, é processada por um conjunto de ações que pode ou não direcionar para a interface seguinte. Assim, para cada atividade temos uma Rede de Interfaces correspondente.

A representação da Rede de Interfaces adota um formalismo visual que informa sobre as interfaces e transições entre elas. O formalismo é análogo ao de *statecharts* [Harel'1988,

Wellner'1989], cujas virtudes para representação de interfaces já foram apregoadas [Shneiderman'1998]. A Figura 12 mostra a Rede de Interfaces da atividade Empréstimo. Cada retângulo representa uma interface que é apresentada para o usuário final. A denominação do retângulo corresponde à identificação da interface. Abaixo da linha divisória mostra-se o conjunto de papéis que têm acesso à interface. Esse conjunto é herdado (parcial ou integralmente) da Rede de Atividades. As setas indicam um conjunto de ações que são realizadas a partir de um acionamento feito por um usuário na interface de origem. As setas são denominadas de acordo com os nomes dos elementos de interface que as ativam e, no caso de haver mais de um destino possível, a denominação é complementada com as condições que determinam esse destino.

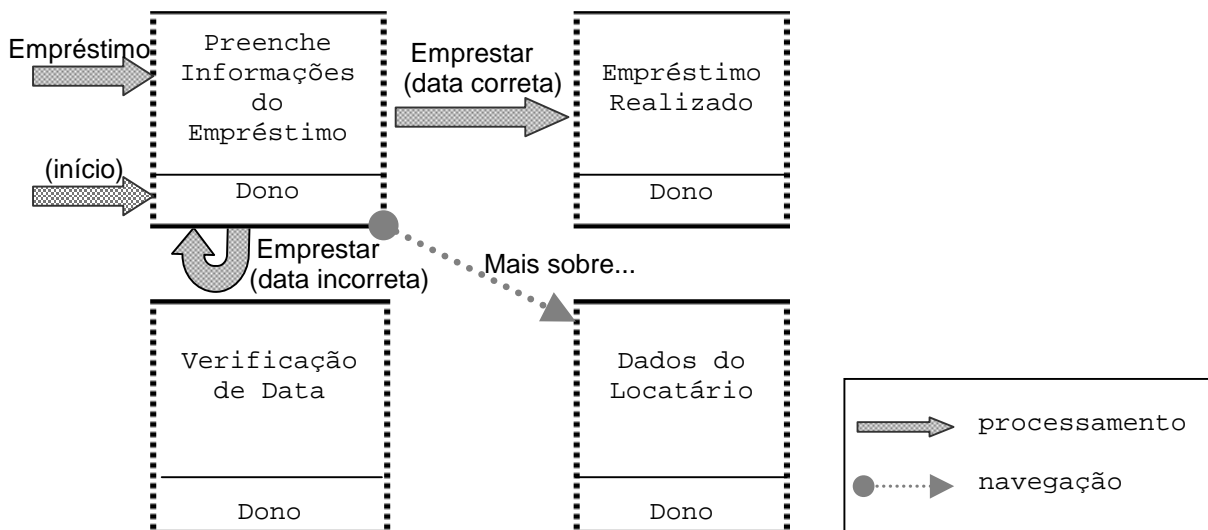


Figura 12. Rede de Interfaces da Atividade Empréstimo

Note-se que há dois tipos de ações representadas por setas diferentes: processamento de informações e navegação. No âmbito desse trabalho definimos navegação como uma ação que visa apoiar o usuário na realização de uma tarefa, mas que não representa um avanço da tarefa no que diz respeito aos modelos previstos e sua implementação em uma aplicação. As navegações apresentam uma interface nova ou modificam uma interface existente provendo informações adicionais para a realização de uma tarefa (e.g. a tomada de uma decisão). Para manter-se o contexto da tarefa enquanto navega-se em busca de mais informações, propomos que as navegações sejam sempre mostradas paralelamente às interfaces de realização de tarefa. No estágio atual só está previsto um nível de navegação. Não estamos tratando ainda

de modelos de navegação mais elaborados, que podem implicar por exemplo em uma rede de navegações. Pretendemos explorar redes de navegação complexas em trabalhos futuros através de uma integração desse trabalho com pesquisas de representação de sistemas hipermídia já existentes, como a proposta da OOHDM [Rossi et al.'1995, Schwabe et al.'1996, Rossi&Schwabe'1998].

No caso da aplicação de Empréstimo implementada, cada uma das interfaces corresponde a uma página *web* de entrada ou saída de informações, compreendendo a utilização de recursos textuais e/ou gráficos e de navegação e preenchimento de formulários. Nesse trabalho não consideramos quaisquer aspectos de multimídia que o ambiente WWW facultaria.

Páginas em HTML que servem somente para criação de sub-janelas, ou seja, que dividem uma janela de *browser* em várias sub-janelas ou *frames*, são representadas na Rede de Interfaces por pontos. A partir desses pontos saem arcos em direção: à cada uma das páginas *web* (de entrada ou saída de informações) de cada um dos *frames*; ou a outro ponto de subdivisão de *frames*. Por exemplo, a janela abaixo está subdividida em dois *frames*: na página do lado esquerdo apresenta-se um menu de opções e na página do lado direito apresenta-se a tela de preenchimento de dados de empréstimo. Essa é a primeira janela aberta quando a aplicação é ativada.

SERG - Empréstimos de Materiais - Microsoft Internet Explorer

File Edit View Favorites Tools Help

>Empréstimo  
 Devolução  
 Material  
 Cadastra  
 Vencimentos

Preencha as informações sobre o Empréstimo.

Material:

Locatário:  Mais sobre...

Data do Empréstimo:  dd/mm/aaaa

Data da Devolução:  dd/mm/aaaa (opcional)

Emprestar

Figura 13. Exemplo do Diálogo “Preenche Informações do Empréstimo”

A Rede de Interfaces apresentará então a seguinte representação (parcial) na qual o círculo preto representa a página HTML que especifica a sub-divisão em *frames*:

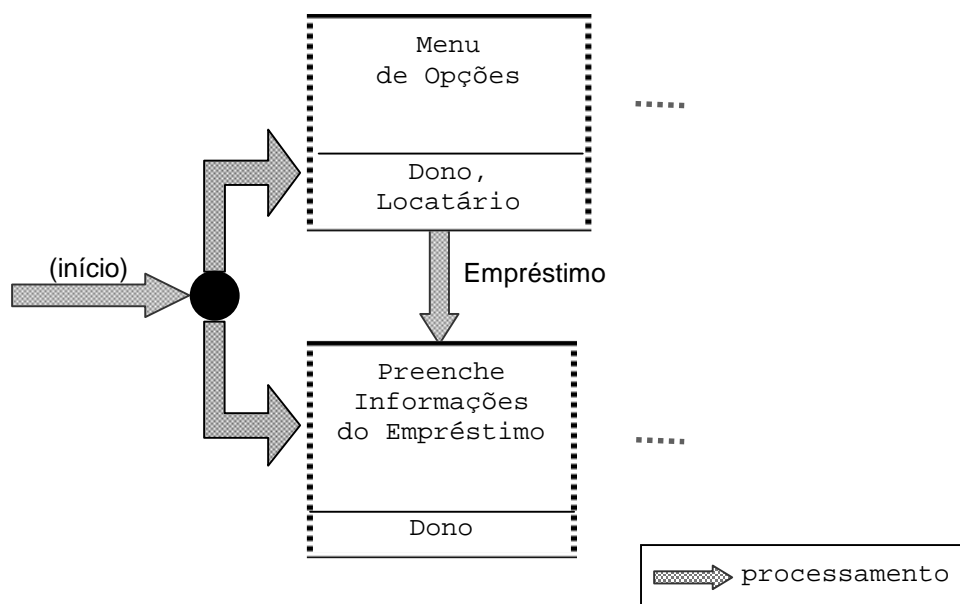


Figura 14. Rede de Interfaces da Atividade Empréstimo

Como veremos posteriormente a representação da Rede de Interfaces pode ser derivada a partir da linguagem formal proposta, aplicando-se os princípios de Contínuo Semiótico e

Abstração Interpretativa. A Rede de Interfaces é importante já que torna viável a construção de extensões através de uma perspectiva da interação e viabiliza também uma comunicação da extensão através dessa perspectiva. Ou seja, ela permite um re-aproveitamento do conhecimento do código da UIL no processo de construção de extensões, cabendo lembrar que essas extensões poderão fazer parte das mensagens que os agentes comunicantes irão trocar entre si.

Como parte do Modelo de Domínio também integrada pela linguagem formal, é necessária uma representação das informações que serão utilizadas durante a realização das atividades previstas e que serão manipuladas durante os passos de interação. Para isso prevemos um Modelo de Informação.

### **Modelo de Informação**

O Modelo de Informação é representado através de modelagem orientada a objetos seguindo a UML [Booch et al.'1999]. A Figura 15 mostra o Diagrama de Classes do domínio de Empréstimos de Materiais.

Cada classe de informação é representada por um retângulo subdividido em duas partes. Na parte de cima mostra-se o nome da classe e na parte de baixo mostram-se os atributos, ou qualificadores, da classe. A Figura 15 mostra uma versão simplificada do Diagrama de Classes, já que não apresenta as operações disponíveis em cada classe (como por exemplo as operações de consulta e atualização de valores dos atributos). As linhas que conectam as classes indicam relações existentes entre elas. Os triângulos indicam que as relações envolvem o conceito de herança, no qual as classes conectadas à base do triângulo (e.g. parte de baixo) herdam os atributos da classe conectada ao topo (vértice oposto) do triângulo. Assim, de acordo com a Figura 15, podemos dizer por exemplo que a classe *Dono* e *Locatário* possuem os mesmos atributos da classe *Pessoa*.

A cardinalidade da relação entre as classes é representada através das notações 0..1, 1, \* ou indicadas nas extremidades das linhas. Portanto, vemos que um locatário pode estar com 1 ou mais (1..\*) materiais. Além disso, os relacionamentos podem ter propriedades próprias. Elas

são representadas por classes de associação cuja notação é a mesma de classes, encontrando-se conectada ao respectivo relacionamento através de uma linha tracejada. Note-se que a relação entre Locatário e Material possui a classe de associação Empréstimo. Essa classe possui dois atributos: a data do empréstimo e a data da devolução.

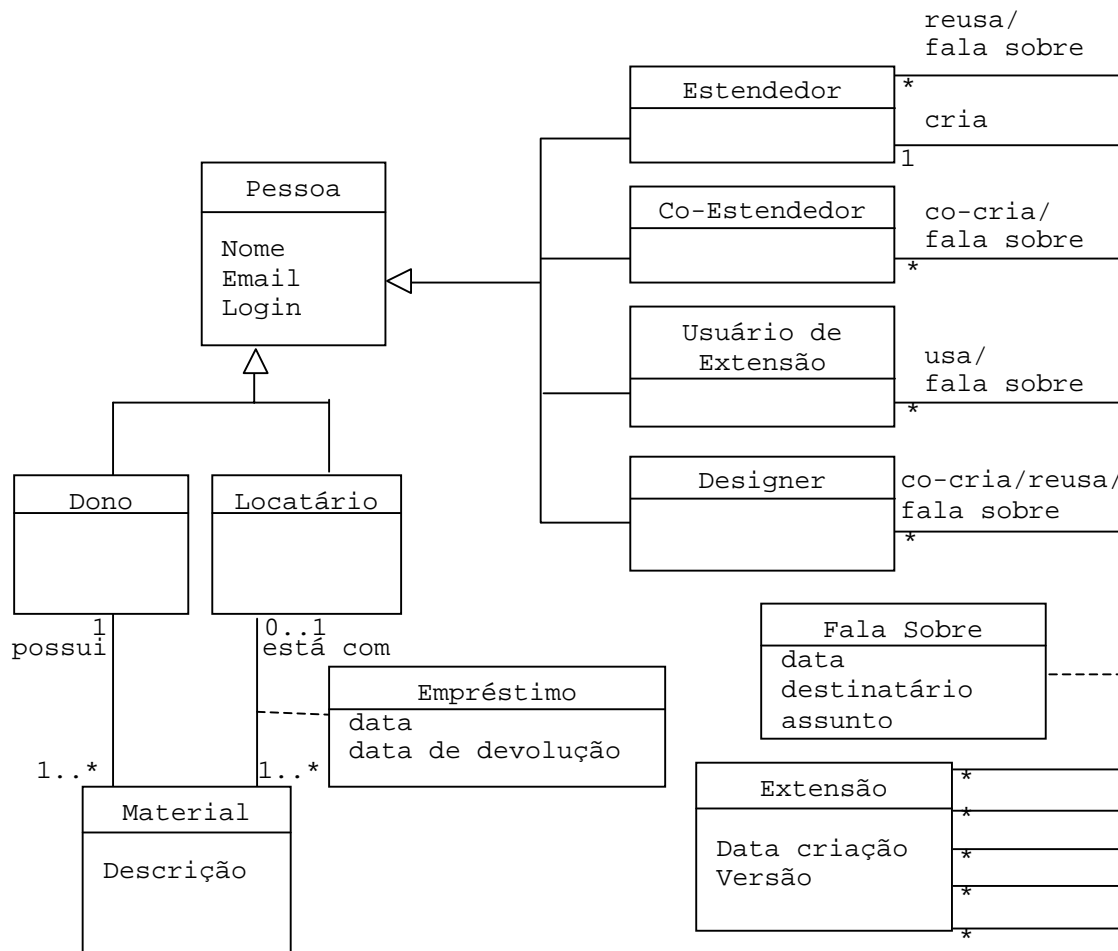


Figura 15. Diagrama de Classes do Domínio de Empréstimo de Materiais

O Modelo de Informação deve ser complementado com representações de seu relacionamento com o Modelo de Interfaces, que será detalhado a seguir. O objetivo é fornecer um mapeamento mais direto entre as informações do domínio e suas unidades expressivas na linguagem de interface. Isso possibilita que usuários e estendedores atuem sobre os códigos de interface para produzir efeitos sobre o domínio da aplicação, podendo também se comunicar utilizando os códigos de interface. Como mencionado em [Cunha et al.'2000a] uma possibilidade é que as classes do Modelo de Informação apresentem, para cada atributo

do domínio, outros atributos ou facetas que descrevam os respectivos recursos expressivos que serão visualizáveis e manipuláveis pelos usuários e estendedores. Por exemplo, podemos relacionar o atributo Descrição (de um Material) com um *widget* de interface para captura de valor (e.g. uma caixa de entrada de texto) e com um outro *widget* para amostragem de valor (e.g. um *label*, ou texto de saída simples). No caso do sistema prever a geração automática de textos, podemos descrever facetas que representem, por exemplo, o gênero (feminino ou masculino) de cada informação do domínio. Para poder-se manipular o conjunto das informações de domínio presentes em um banco de dados, é preciso que sejam descritas facetas que representem, por exemplo, o nome da tabela e da coluna do banco de dados onde uma informação do domínio se encontra armazenada.

### **Modelo de Grupo**

O Modelo de Grupo representa o relacionamento entre os membros de um grupo e entre eles e os objetos do domínio. Para esse modelo estamos seguindo a proposta de [Prates'1998, Prates&de Souza'1998], na sua versão estendida por [Dahis'2001]. É importante notar que é necessário para a solução que estamos propondo aqui representar o designer da aplicação como membro do grupo e as extensões como objetos da aplicação.

Primeiramente definiremos os papéis dos usuários, instanciados para o uso da aplicação de Empréstimo de Materiais. Como representado no Diagrama de Classes da Figura 15 temos os papéis de Dono e Locatário de Material. Gostaríamos de complementar o diagrama para dar conta do desenvolvimento de extensões, adicionando os papéis de: Estendedor, Co-Estendedor, Usuário de Extensão e Designer. A Figura 16 mostra o conjunto integral de papéis da aplicação. Note-se que segundo o modelo um usuário pode, por exemplo, exercer o papel de Dono de um material e, ao mesmo tempo, de Estendedor da aplicação. Em outra perspectiva, cabe lembrar que um usuário também pode exercer, ao mesmo tempo, o papel de Dono de um material e o papel de Locatário de um outro material de um outro Dono.



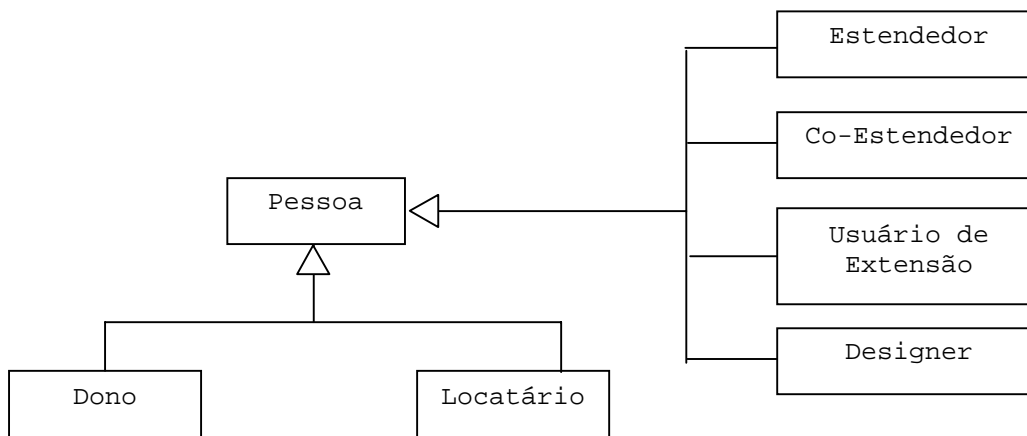


Figura 16. Papéis no Modelo de Grupo da Aplicação de Empréstimo de Materiais

Nesse domínio, não há relação hierárquica entre os papéis. Ainda assim, pode haver uma relação de dependência entre eles em determinadas ocasiões. Por exemplo, pode ocorrer a situação onde o estendedor depende do designer ou de um co-estendedor para conseguir construir totalmente uma extensão, como no caso onde o estendedor não tem acesso a um objeto necessário para a efetivação da extensão.

Para a aplicação de Empréstimos de Materiais, há dois modelos de colaboração vigentes entre os (papéis dos) membros do grupo. Durante a utilização da aplicação, temos que Donos e Locatários de materiais colaboram de acordo com o modelo de colaboração chamado Encaixe Nebuloso [Prates'1998], onde o trabalho de um usuário afeta o de outro (e.g. a data que o Dono especificar como data de devolução será a data quando o Locatário deverá devolver o material), mas a integração entre essas tarefas é definida pelos usuários através de um protocolo social [Ellis et al.'1991], ou seja, de forma não prescrita pela aplicação.

Já durante a extensão da aplicação, ocorre o modelo de colaboração chamado de Sobreposição [Prates'1998]. Nesse modelo, os membros do grupo (usuários e designers) possuem tanto trabalhos individuais quanto compartilhados. Ou seja, eles podem trabalhar individualmente,

interagir com outros membros diretamente e atuar sobre o contexto de forma que impacte ou não o trabalho dos outros membros, como no caso da construção colaborativa de extensões.

Concluindo o modelo de colaboração, devemos destacar a observação de que, se um membro precisa mudar de um modelo para outro freqüentemente, ele está sujeito a ficar confuso [Prates' 1998, Prates&de Souza' 1998]. Em um primeiro momento, estamos lidando com esse cenário através de uma distinção clara entre os ambientes de utilização e o de extensão de forma que, com isso, o membro do grupo tenha indicação sobre o modelo de colaboração vigente a cada momento e não se confunda. A efetividade dessa solução precisará ser cuidadosamente avaliada em trabalhos futuros, inclusive tratando-se sobre se a mesma é aplicável a outros domínios que tenham outro conjunto de modelos de colaboração.

Para finalizar o Modelo de Grupo, apresentamos tabelas que mostram os objetos da aplicação na primeira linha e depois mostram os tipos de ação que os membros do grupo têm sobre esses objetos.

| <b>Material</b> |             |     |             |
|-----------------|-------------|-----|-------------|
|                 | Atuar sobre | Ver | Falar sobre |
| Dono            | X           | X   | X           |
| Locatário       |             |     |             |

| <b>Material Emprestado</b> |             |     |             |
|----------------------------|-------------|-----|-------------|
|                            | Atuar sobre | Ver | Falar sobre |
| Dono                       | X           | X   | X           |
| Locatário                  |             | X   | X           |

| <b>Extensão</b>               |             |     |             |
|-------------------------------|-------------|-----|-------------|
|                               | Atuar sobre | Ver | Falar sobre |
| Estendedor                    | X           | X   | X           |
| Co-Estendedor                 | X           | X   | X           |
| Designer como (co-)estendedor | X           | X   | X           |
| Usuário da Extensão           |             | X   | X           |

Figura 17. Objetos da Aplicação e Atuação dos Membros do Grupo

De acordo com as consistências propostas nos trabalhos citados, nota-se o fato de que um locatário em potencial não tem nenhum conhecimento sobre os materiais cadastrados e que isso poderia ser interpretado como uma exclusão do objeto Material do escopo de trabalho dos Locatários. No entanto, isso não é desejável, tendo em vista que locatários irão pedir materiais emprestados aos seus donos. O motivo para tal decisão de design foi o de incentivar a criação de extensões, de forma a se garantir uma avaliação da proposta. Assim, a aplicação foi desenhada e implementada em uma versão bastante simplificada. Como veremos no capítulo de avaliação, essa estratégia surtiu o efeito esperado e uma das sugestões de extensão foi a de criação de acesso de consulta a todos os materiais por parte dos possíveis locatários.

Note-se também que foi modelado um alto grau de colaboração sobre o objeto Extensão. A necessidade dessa colaboração já foi justificada anteriormente e indica a importância de um ambiente de comunicação para apoio à mesma. Nesse trabalho estamos atacando os níveis de interação individual e interpessoal.

## 4.2. Modelo de Interfaces

O Modelo de Interfaces é subdividido em duas partes. Primeiramente é necessário detalhar a Rede de Interfaces representada no Modelo de Tarefas. Para isso, deve-se especificar as possíveis interações entre usuário e sistema. O ideal é que essa especificação seja feita de forma independente à tecnologia de implementação para que permaneça como uma representação estável da mensagem do designer sobre como se dará a conversa entre o usuário e seu porta-voz: o sistema. Para esse fim, [Dahis'2001] propôs uma Linguagem para Especificação de Cenários de Interação (LECI) que é “uma ferramenta para auxiliar designers a especificar instâncias de modelos de interação”.

O trecho de código a seguir mostra a utilização da LECI para a descrição do diálogo `Preenche Informações do Empréstimo`, que faz parte da Rede de Interfaces da atividade `Empréstimo` mostrada na Figura 12. Esse diálogo se refere ao *frame* da direita da tela apresentada na Figura 13.

```
Role Pessoa may access Cadastro de Material, Vencimentos
Role Dono sub-role of Pessoa may access Empréstimo, Devolução

Concept identificação_do_dono type Introduced by Technology
Concept identificação_do_locatário type Introduced by Technology

Concept lista_de_materiais_disponíveis_de_um_dono type
  Introduced by Technology
Concept identificação_do_material type Introduced by Technology
Concept lista_de_locatários type Introduced by Technology
Concept data_de_empréstimo type Original Domain
Concept data_atual type Original Domain
Concept data_de_devolução type Original Domain
```

```

shared Task Empréstimo (Creation)
operands identificação_do_dono
Sequence
{
  Exclusive-Contexts
  {
    When context is Interaction-Context(dono_tem_material,
      {identificação_do_dono, lista_de_materiais_disponíveis_de_um_dono},
      <a lista_de_materiais_disponíveis_de_um_dono não está vazia>,
      {Dialog Preenche Informações do Empréstimo (Domain Information)
        Dialog Confirmação do Empréstimo (Operation Confirmation) }

    When context is Interaction-Context(dono_não_tem_material,
      {identificação_do_dono, lista_de_materiais_disponíveis_de_um_dono},
      <a lista_de_materiais_disponíveis_de_um_dono está vazia>,
      {Dialog Não há Material para Emprestar (Feedback)})
  }
}

```

```

shared Dialog Preenche Informações do Empréstimo(Domain Information)
operands identificação_do_dono
sequence
{
  system: <Preencha as informações sobre o Empréstimo.> (Explicative Help)
  subtopic <material>
  {
    system: <Material:> (Information Request)
    user alternatives <lista_de_materiais_disponíveis_de_um_dono>
    user chooses mandatorily: identificação_do_material (Domain Information)
  }
  subtopic <locatário>
  {
    system: <Locatário:> (Information Request)
    user alternatives <lista_de_locatários>
    affords: dono inicializes dados_do_locatário causing
      persistence (Detailed Information Request)
    user chooses mandatorily: identificação_do_locatário (Domain Information)
  }
  subtopic <data de empréstimo>
  {
    system: <Data de Empréstimo:> (Information Request)
    suggestion data atual
    system: <dd/mm/aaaa> (Preventive Help)
    user informs mandatorily: data_de_empréstimo (Domain Information)
  }
  subtopic <data de devolução>
  {
    system: <Data de Devolução: (opcional)> (Information Request)
    system: <dd/mm/aaaa> (Preventive Help)
    user informs optionally: data_de_devolução (Domain Information)
  }
  dono inicializes empréstimo_realizado causing termination (Confirmation)
}

```

```

has user errors handled by
  handle_user_errors(immediate, interactive correction, autonomous)

  When context is Interaction-Context( data_inválida ,
    {data_de_empréstimo},
    (< mês < 1 ou mês > 12 ou dia_inválido ou ano < 2001 > )
    { Dialog Verificação de Data (Feedback) } )

  When context is data_inválida ,
    {data_de_devolução},
    (< mês < 1 ou mês > 12 ou dia_do_mês_inválido ou ano < 2001 > )
    {Dialog Verificação de Data (Feedback)} )

  When context is Interaction-Context( data_de_devolução_inválida ,
    {data_de_devolução},
    (<data_de_devolução < data_de_empréstimo > )
    { Dialog Verificação de Data (Feedback)} )

```

Para uma solução prática, a especificação na LECI deve ser complementada com a representação dos recursos expressivos que serão utilizados durante a interação usuário-sistema, como os presentes na Figura 13 (elos de navegação, campos para entrada de texto, botões de acionamento, etc.). Ou seja, é preciso representar também o conjunto de *widgets* de interface disponíveis, que será dependente do ambiente de implementação.

Para uma primeira avaliação da nossa proposta, escolhemos o ambiente da WWW e dos navegadores Internet Explorer da Microsoft e Netscape Navigator da Netscape. A opção por esse ambiente apresenta vantagens práticas. Primeiramente destacamos a disponibilidade conjunta dos protocolos HTTP, FTP e de E-mail da Internet que provê condições benéficas para uma integração entre o ambiente de uso do software e o de comunicação dentro de uma mesma aplicação. Outro ponto positivo é a possibilidade de alcançar usuários distribuídos geográfica e temporalmente. Adicionalmente, o uso desse ambiente é relativamente popular entre usuários da Internet e durante sessões de avaliação pode-se dispensar um treinamento nesse sentido e permitir a concentração do foco na nossa proposta.

Portanto, foi inicialmente considerado o conjunto de *widgets* previsto na linguagem HTML (*Hypertext Mark-up Language*) para expressão de formulários e elos de navegação, além de textos formatados. Lembramos que nossa proposta não trata dos recursos de multimídia disponíveis no ambiente WWW. O conjunto considerado foi complementado com os recursos de interação providos pelas linguagens JavaScript e JScript de processamento na máquina cliente.

Atentou-se também para os padrões e restrições que já são conhecidos deste ambiente. Por exemplo, os usuários dos navegadores mencionados estão acostumados com a possibilidade de retornar à página *web* imediatamente anterior à atual, de “cancelar” a continuação de um processamento (não esperando por uma resposta de uma requisição HTTP já feita), ou de desativar o interpretador da linguagem JavaScript ou JScript. A aplicação de workflow foi planejada e implementada prevendo essas situações e formas de lidar com elas para garantir seu funcionamento adequado.

Nesse capítulo vimos que através da construção de modelos o designer especifica a mensagem que deseja transmitir aos seus usuários sob diferentes perspectivas e em diferentes graus de abstração. A Figura 18 mostra uma visão abrangente dos modelos sugeridos:

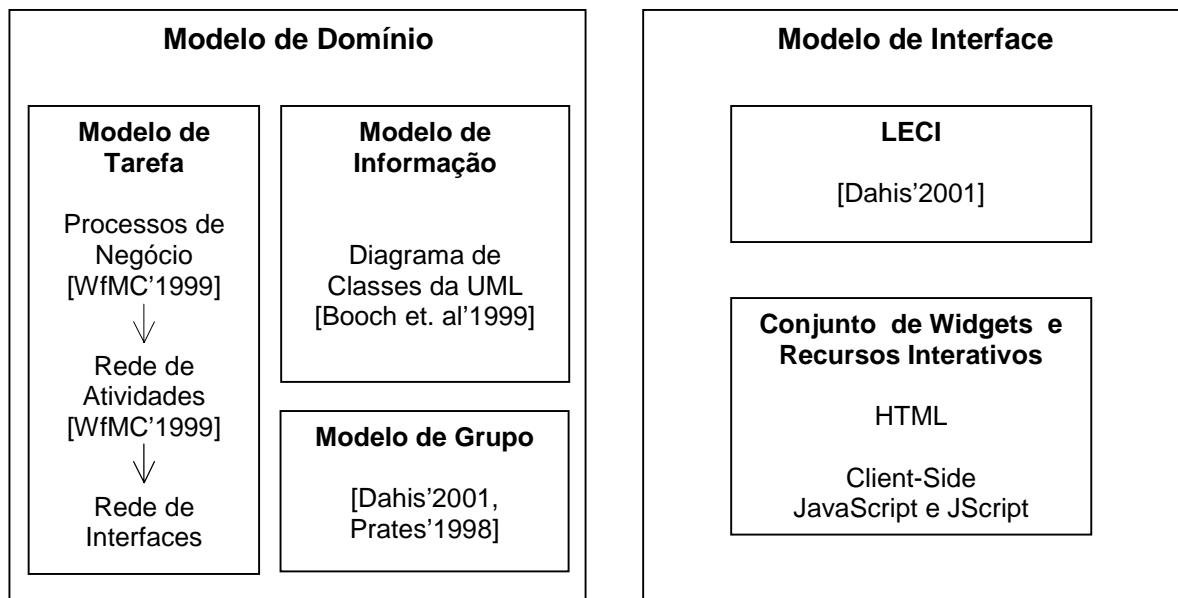


Figura 18. Modelos Previstos

Pode-se argumentar que o custo do processo de elaboração de todos esses modelos é alto, mas é esse processo que guia o designer para uma visão ampla e ao mesmo tempo profunda sobre a mensagem a ser transmitida para os usuários finais. Cabe notar que há interseções entre algumas informações representadas nos diferentes modelos. Por exemplo, os papéis dos membros do grupo estão representados nos modelos de Grupo, Informação e de Interface. No entanto, a redundância de representação é apenas superficial, dado que em cada modelo serão ressaltadas diferentes perspectivas e relações de uma mesma informação. Para cada papel dos

membros do grupo especificam-se: os seus atributos (no Modelo de Informação), a inexistência de relação hierárquica entre os papéis e relações com objetos do domínio (no Modelo de Grupo) e os controles de acesso a diálogos de interação (no Modelo de Interface).

Ainda assim, seria interessante estudar em trabalhos futuros maneiras de tornar mais eficiente e menos custosa a construção de todos esses modelos, tal como a criação de ambientes integradores de ferramentas para especificação dos mesmos.

O passo seguinte à construção da mensagem abstrata do designer é a transformação da mesma em um artefato computável, que interage diretamente com o usuário final como representante do designer, que é extensível e sobre o qual possam ser realizadas comunicações. Essa transformação passa necessariamente pela formalização da mensagem abstrata. Usualmente isso se dá a partir de um processo de programação. Nesse trabalho propomos uma formalização através de uma linguagem cujo intuito de design é apoiar também, e principalmente, as comunicações envolvidas nos processos evolutivos da mensagem original, onde os usuários finais (ou um subgrupo deles) integram esses processos.

### **4.3. X-DIS: Linguagem de Representação de Discursos**

A linguagem formal de representação e extensão expressa as interações relativas às tarefas do domínio através de uma *Augmented Transition Network* (ATN) [Woods'1970]. Ela é um processador da Rede de Interfaces e atua como integradora dos modelos descritos. Seu conceito é análogo ao de *scripting languages* que são projetadas para *gluing*: “linguagens que assumem a existência de um conjunto de poderosos componentes e que são destinadas primariamente a conectar componentes” [Ousterhout'1998]. Assim apresentamos uma linguagem cujo poder não está somente na sua equivalência à máquina de Turing, mas também na sua integração de outras linguagens ou perspectivas, aspecto ainda pouco explorado por exemplo nas linguagens de programação por demonstração [McDaniel'2001].

Ela é também uma base referencial comum para linguagens mais abstratas que podem ser derivadas a partir dela, tais como as linguagens dos ambientes de uso e de extensão, linguagens para geração automática de discursos pseudo-naturais que, por exemplo,



sumarizem planos de realização de tarefas (tanto em termos gerais quanto de forma instanciada de planos que foram executados no passado), etc. Atuando como referencial comum para diferentes abstrações, a linguagem pode ser considerada como integrante da base do conhecimento compartilhado entre usuários e designers da aplicação.

Na representação da linguagem formal em ATN, cada estado do nível mais alto/externo da rede indica um passo de interação com o usuário. Essa linguagem formal está descrita de forma que ao se derivar a UIL a partir dela garante-se o Ciclo Mínimo de Interação e, potencialmente, o Contínuo Semiótico com a UIL. A Figura 19 mostra um gráfico abstrato de como a rede de transição de estados da linguagem formal garante o Ciclo Mínimo de Interação.

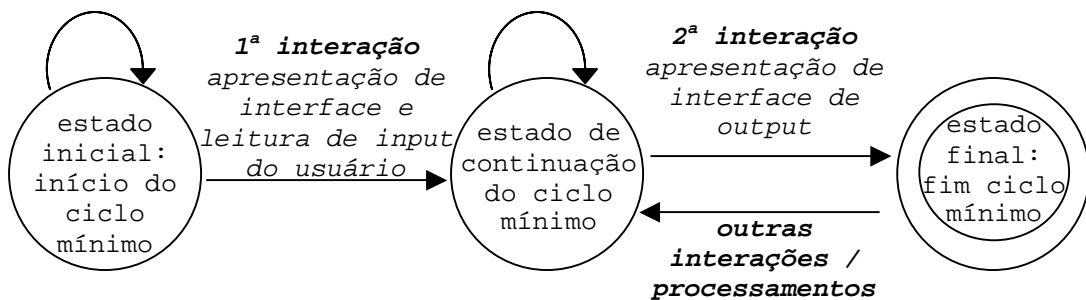


Figura 19. Rede Abstrata de Transição de Estados da Linguagem Formal

A ATN possui uma parte fixa, independente de domínio (representação de estados e transições, correspondentes à passos de interação) e uma parte dependente do domínio (estrutura dos registradores, layout de interfaces, etc. que são normalmente as referências para os elementos dos modelos descritos). A linguagem pode ser adaptada desde que se mantenha a característica de rede de transição de estados que garante o ciclo mínimo de interação. A seguir são descritas as regras de transição da ATN, mostradas na forma de regras de reescritura. A simbologia é explicada após a descrição das regras.

```
rede_de_interfaces → arcset arcset*
arcset → $estado_inicial$ arcos_iniciais*
/* prepara informações antes do ciclo mínimo até que se dá o primeiro passo
de interação: a aplicação apresenta uma tela de entrada ou saída de
informação e lê um input feito pelo usuário */
```

```

        $continua_ciclo_mínimo$ arcos_cont_ciclo*
/* continua no processamento até que se mostra uma segunda tela e se lê
acionamento feito pelo usuário. Com isso fecha-se o ciclo mínimo e passa-se
ao estado final */

```

```

        $fim_ciclo_mínimo$ arcos_cont_ciclo*
/* termina-se a interação ou, caso ainda haja arcos a serem seguidos, dá-se
continuidade ao ciclo mínimo. Isso garante que não são feitos
processamentos sem que depois se apresente algum tipo de informação ao
usuário */

```

```

arcos_iniciais → 'cat' 'data_entry' test* action load_input_fields
                generate_input_screen (read_input_fields action*)+
                read_trigger terminate_cont_ciclo |
'cat' output_screen_type test* action generate_output_screen
                read_trigger action terminate_cont_ciclo |
'push' to_state test* action terminate_inicial |
'tst' label test* action generate_output_screen read_trigger action
        terminate_inicial |
'tst' label test* action terminate_inicial |
'pop' form test* |
'parallel' terminate_inicial terminate_inicial+ [monitor]
'parallel' action generate_frames terminate_inicial terminate_inicial+
        [monitor]

```

```

arcos_cont_ciclo → 'cat' 'data_entry' test* action load_input_fields
                  generate_input_screen (read_input_fields action*)+
                  read_trigger terminate_cont_ciclo |
'cat' output_screen_type test* action generate_output_screen
                  read_trigger action terminate_fim |
'push' to_state test* action terminate_cont_ciclo |
'tst' label test* action generate_output_screen read_trigger action
        terminate_fim |
'tst' label test* action terminate_cont_ciclo |
'pop' form test* |
'parallel' terminate_cont_ciclo terminate_cont_ciclo+ [monitor]
'parallel' action generate_frames terminate_cont_ciclo
        terminate_cont_ciclo+ [monitor]

```

```

output_screen_type → 'output_message' | 'object_visualization' | 'menu'

test → 'security_id_validation' | 'access_control' | 'validate_input' |
       'arbitrary_pre_condition' | 'synchronize' | 'communicate' | 'monitor'

action → 'setr' register form |
        'sendr' register form |
        'liftr' register form |
        'setdb' db_oper dbitem form |
        action*

generate_output_screen → 'setr' register 'generate_htmltext' [#parameters#] |
                        'setr' register 'visualize' [#parameters#]

load_input_fields → setr %interface_input_fields% load_interface_structure

generate_input_screen → 'setr' %interface_name% quote 'generate_htmlform'
                       %interface_input_fields% [other_params]

read_input_fields → 'setr' %interface_input_fields% 'buildq'
                   #interface_structure# #read_input# |
                   'sendr' %interface_input_fields% 'buildq' #interface_structure#
                   #read_input# |
                   'liftr' %interface_input_fields% 'buildq' #interface_structure#
                   #read_input# |
                   read_input_fields*

/* lê input de trigger da interface: botão, link ou fechamento da janela */
read_trigger → 'setr' %trigger_register% #read_trigger# |
              'sendr' %trigger_register% #read_trigger# |
              'liftr' %trigger_register% #read_trigger#

register → #tmpvariable# | #navigational_item# | #nl_text_structure_item#

form → 'getr' register |
       'getdb' dbitem |
       #fixed_value# | /* valor constante */
       'buildq' #structure# register* form |
       'quote' executable_code #parameters#

```

```

load_interface_structure -> 'buildq' #interface_structure# register* form

db_oper ->  'insert' | 'update' | 'delete'
           /* a operação de leitura é feita pelo form 'getdb' */

dbitem ->  'table' #table# {'field' #field#}*
           ['where_clause' #parameters#] |
           /* pode-se mencionar um ou mais campos (fields) da tabela e a
           cláusula where é opcional */

executable_code -> 'destroy' #object#      |
                  'substitute' #object#    |
                  'multiply' #object#      |
                  'generate_nltxt' #parameters# |
                  'generate_navigational_anchor' #origem# #destino#

terminate_inicial ->  'to' $estado_inicial$ post_test*

terminate_cont_ciclo -> 'to' register post_test* | /*register armazena o
nome do estado, que deve ser um estado de continuidade do ciclo mínimo ou
final */
                   'to' $continua_ciclo_mínimo$ post_test*

terminate_fim ->  'to' register post_test* | /*register armazena o nome do
estado, que deve ser um estado de continuidade do ciclo mínimo ou final */
                'to' $continua_ciclo_mínimo$ post_test*
                'to' $fim_ciclo_mínimo$ post_test* /* fim */

post_test ->  'arbitrary_post_condition'

```

### Simbologia

- |      representa maneiras alternativas para a construção
- \*      indica zero ou mais repetições
- +      indica uma ou mais repetições
- #      itens delimitados por este símbolo referem-se a símbolos terminais ou designam produções que não precisam ser detalhadas para o enfoque desse trabalho

- ⌘ itens delimitados por este símbolo indicam nomes de variáveis
- \$ itens delimitados por este símbolo indicam que estados da rede de transição e o seu tipo, de acordo com a rede abstrata apresentada na Figura 19.

### Tipos de Arcos

`cat` arco que pode ser seguido caso o símbolo de `input` seja um membro de `output_screen_type` ou igual a `data_entry` e que o teste (caso haja) seja satisfeito. Somente arcos `cat` geram páginas de interface HTML relacionadas ao workflow. A estrutura básica de um arco `cat` é: realização de testes e ações, construção de estruturas relacionadas à página HTML, apresentação da página HTML, leitura (de dados e) de uma ação terminal (trigger), passagem de controle ao próximo estado.

`push` e `pop` arcos que transferem entre redes de transição em uma pilha. O arco `pop` indica em que condições um estado é considerado final e o `form` deve ser retornado como valor da computação, caso o arco `pop` seja escolhido.

`tst` é um arco que executa testes internos da aplicação e, caso os testes falhem, gera página com mensagens de erro. Ou seja, este é um arco para tratamento de erro.

### Ações

`setr` atribui a um registrador do nível atual da pilha o valor de `form`. O registrador armazena informação que é construída e que se quer manter a medida que se transita de um estado para o outro

`sendr` atribui a um registrador do nível seguinte (abaixo) da pilha o valor de `form`

`liftr` atribui a um registrador do nível seguinte (acima) da pilha o valor de `form`

`setdb` atualiza uma tabela de banco de dados de acordo a operação especificada (`db_oper`), inserindo, atualizando ou apagando os dados dos campos mencionados

`getdb` retorna o valor de um ou mais campos de uma tabela

`to` indica que se deve passar ao próximo estado, que pode ser pré-determinado, ou estabelecido a partir do valor de um registro. Caso haja alguma pós-condição, só se passará ao próximo estado caso essa seja satisfeita

`form` assim como o símbolo `action`, representa uma chamada de função onde o primeiro elemento é o nome de uma função e os restantes representam parâmetros da função

`test` condições a serem satisfeitas, também representadas como chamadas de função

`getr` é uma função cujo valor é o conteúdo do registrador indicado

`buildq` é um construtor de estruturas. Toma uma estrutura de campos marcados e retorna a mesma estrutura com campos preenchidos de acordo com os registradores indicados. Note-se que essas estruturas são construídas de acordo com a representação de informação especificada para o domínio (exemplo: uma instância da classe `Material` com alguns atributos preenchidos). Dessa forma são manipulados os elementos dos modelos que a linguagem integra

`quote` chamada a uma função arbitrária que retorna um valor para `form`

O trecho de texto a seguir mostra a especificação em ATN do passo de interação `Preenche Informações do Empréstimo` (que foi mencionado na Figura 12 e na Figura 13).

```
cat(data_entry, /* tela de entrada de dados */
    test(access_control), /* checa se usuário é dono de material */
    (
        /* coloca em uma estrutura os dados para a interface, não só os dados
        provenientes de banco de dados, mas também, caso haja, dados de uma
        interação ou processamento anterior, inclusive mensagem de erro */
        load_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Material getdb[table:'Material' field:'Descr']),

        load_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Nome_Loc getdb[table:'Locatário' field:'Nome']),

        load_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Data_Empréstimo quote 'obtem_data_atual'),

        load_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.actions quote 'obtem_actions'
Interface_Cad_Empréstimo),

        load_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.anchors quote 'generate_navigational_anchor'
Interface_Cad_EmpréstimoOO.Nome_Loc 'Mais sobre...'),

        /* gera formulário */
        generate_input_screen(setr 'Preenche_Empréstimo' 'quote generate-
htmlform' 'WebPage01'),

        /* lê inputs do usuário, armazenando-os em registradores */
        read_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Material_Escolhido read_input),
```

```

    read_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Nome_Loc_Escolhido read_input),

    read_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Data_Empréstimo read_input),

    read_input_fields(setr 'WebPage01' buildq
Interface_Cad_EmpréstimoOO.Data_Devolução read_input),

    /* lê a ação escolhida pelo usuário (processamento ou navegação) */
    read_trigger(setr 'SelectedTrigger' read_trigger)
)

/* transfere (ou navega, possivelmente em paralelo) para o próximo
estado, de acordo com a ação escolhida pelo usuário (processamento ou
navegação) */
    terminate_cont_ciclo(to SelectedTrigger, post_test)
)

```

Figura 20. Sentença da X-DIS: Preenchimento de Dados de Empréstimo

O Apêndice I mostra integralmente o código de implementação da representação formal.

## 5. Avaliação Empírica do Trabalho

Tendo descrito o Modelo Semiótico dos Processos Comunicativos relacionados à Atividade de Extensão e a Linguagem para Representação de Discursos sobre Extensões, passamos agora para a descrição de uma avaliação desse trabalho. Cabe notar que não pretendemos dizer que validamos nosso modelo, já que a validação de um modelo para design requer que se construa uma série de implementações de produtos gerados a partir deste modelo e que se avalie, pela qualidade do produto, a qualidade do modelo de design, supondo-se um designer ideal, que não introduz erros na passagem do modelo conceitual para o artefato implementado. Uma validação exigiria também a existência de uma correspondência comprovável entre um design e um produto implementado e, mais ainda, uma identificação de como é feita esta correspondência em modelos parciais de design. Somente o estudo sistemático dessa correspondência já exigiria um conjunto de trabalhos de longa duração, como estudos de casos distintos e/ou a construção de esquemas conceituais que explicitam como se dá a correspondência. Portanto, uma validação do modelo está fora do nosso escopo, embora trabalhos futuros nesse sentido devam ser perseguidos de forma a melhor compreender os resultados e efeitos da utilização do modelo na construção e uso de sistemas extensíveis.

A avaliação apresentada nesse capítulo constitui uma investigação empírica realizada com usuários no âmbito de uma aplicação do domínio de Empréstimos de Materiais. A investigação é guiada pelo modelo proposto e mostra como ele auxilia na realização de previsões e explicações sobre os fenômenos comunicativos focalizados nesse trabalho. A investigação mostra também como uma aplicação do modelo pode compor uma referência descritiva para especificação, implementação, discussão e revisão de projetos de sistemas extensíveis que venham a ser desenvolvidos com base nele. Assim, mostramos o papel do modelo no processo de avaliação formativa [Preece et al.'1994] que influencia o desenvolvimento de uma aplicação.

Baseados nessa investigação, fazemos uma avaliação preliminar da linguagem X-DIS comentando sobre sua expressividade e sobre linguagens que poderiam ser derivadas dela, tais



como a EUPL e a UEL. Finalmente apontamos para algumas soluções que poderiam integrar sistemas extensíveis que fossem baseados na X-DIS.

Para a investigação empírica apresentou-se a um grupo de usuários uma aplicação computacionalmente implementada, não extensível, do domínio de Empréstimos de Materiais. Foi pedido que os usuários a utilizassem para realizar empréstimos, devoluções de materiais, etc. familiarizando-se assim com o código da UIL da aplicação. Depois eles foram incentivados a imaginar extensões a essa aplicação e a documentá-las em papel, com vistas a uma comunicação assíncrona com a designer da aplicação.

Para a documentação foram disponibilizadas, também em papel, representações abstratas para serem utilizadas como base expressiva, incluindo-se aí representações da UIL (e.g. *dumps* de tela), já que não se espera que os usuários se expressem diretamente através da X-DIS. O registro de extensões em papel foi permitido com o objetivo de não limitar a expressão dos usuários à X-DIS. Assim pôde-se avaliar as restrições que a linguagem pode vir a introduzir quando utilizada para expressão de extensões. Durante a análise dos resultados discutiremos mais detalhadamente esse aspecto.

## **5.1. Preparação e Realização do Teste com Usuários**

### **Domínio da Aplicação: Empréstimo de Materiais**

Algumas informações sobre esse domínio foram dadas no capítulo 4, onde descreveu-se a X-DIS e os modelos que ela integra. As principais atividades previstas na aplicação implementada foram esquematizadas na Figura 11 e as reescrevemos aqui: Empréstimo de Material, Devolução de Material, Cadastramento de Material e Visualização de Vencimentos. Esta última permite que um usuário veja os materiais que deve devolver e que devem ser devolvidos a ele com, caso haja, suas respectivas datas de devolução previstas.

## Implementação da Aplicação

Para motivar a criação de extensões e a conseqüente comunicação sobre as mesmas, a aplicação foi implementada de forma deliberadamente sumária, contendo apenas funcionalidades básicas que garantissem uma forma de realização das atividades mencionadas. A seguir detalhamos a implementação dessas atividades.

### Empréstimo de Material

Essa atividade foi descrita ao longo desse trabalho nas seções que compõem o capítulo de descrição da X-DIS. A Figura 12 mostra a rede de interfaces que compõe a atividade. Cabe destacar que essa atividade só pode ser realizada pelo dono do material que está sendo emprestado. Ou seja, o locatário não pode registrar um empréstimo feito a ele e também o dono de um outro material não pode registrar um empréstimo de um material que não lhe pertence.

A Figura 13, na seção 4.1, mostrou a interface de preenchimento de empréstimo de material. Antes de confirmar um empréstimo, pode-se obter informações adicionais sobre o locatário, como exemplificado na interface a seguir. Note que o campo Data de Devolução não apresenta valor correspondente. Isso é efeito do fato de que o seu preenchimento não é obrigatório durante o registro do empréstimo. A decisão pela não obrigatoriedade refletiu a forma de trabalho desse grupo de usuários.

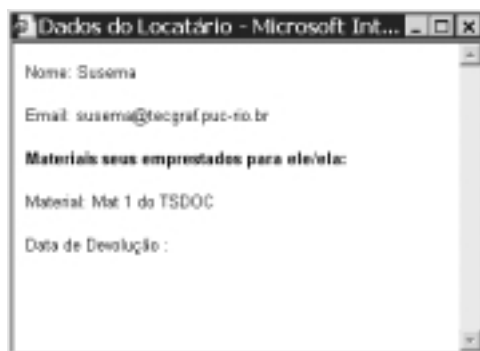


Figura 21. Interface de Informações Adicionais sobre o Locatário

Depois de informados os dados sobre o empréstimo, faz-se uma crítica sobre a validade das datas informadas. Se alguma das datas for inválida (e.g. 33/01/2001), apresenta-se uma mensagem de erro.



Figura 22. Interface de Mensagem de Erro de Data Inválida

Caso todos os dados estejam corretos, o empréstimo é realizado e apresenta-se a seguinte interface de confirmação de empréstimo:



Figura 23. Interface de Confirmação de Empréstimo

## **Devolução de Material**

A rede de interfaces desta atividade é apresentada a seguir:

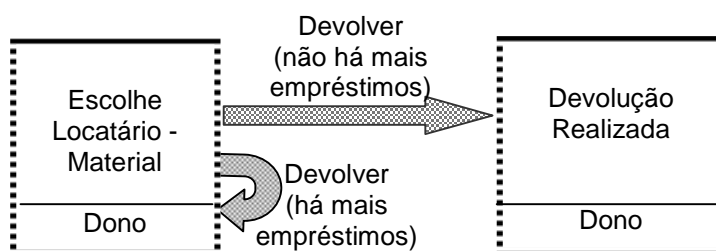


Figura 24. Rede de Interfaces de Devolução de Material



Na primeira interface, o dono do material escolhe o empréstimo que está sendo devolvido:

Figura 25. Interface de Devolução de Material

Caso haja mais empréstimos a serem devolvidos, retorna-se a essa mesma interface, apresentando-se abaixo do botão “Devolver” uma mensagem de confirmação da devolução anterior. Caso não haja mais empréstimos a devolver, apresenta-se ao dono do material a seguinte interface:

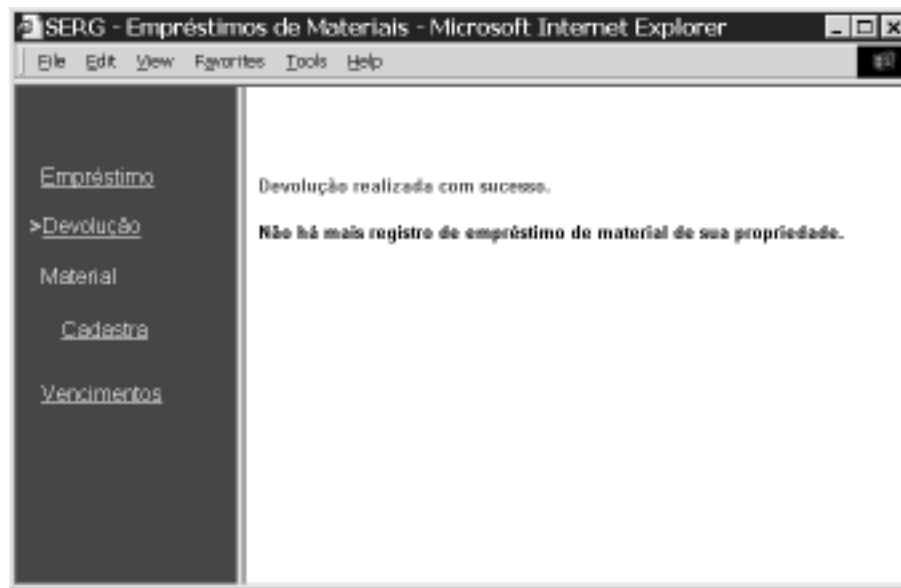


Figura 26. Interface de Confirmação de Devolução

### Visualização de Vencimentos

Um usuário, seja ele um locatário ou um dono de material, pode consultar os empréstimos que ele deve devolver e os empréstimos que devem ser devolvidos a ele, com suas respectivas datas de devolução previstas, como mostrado no exemplo a seguir, no qual o usuário é, ao mesmo tempo locatário de um material e dono de um outro que está emprestado. Novamente lembramos que as datas de devolução não estão preenchidas pois a aplicação não exige um preenchimento das mesmas quando da realização do empréstimo.

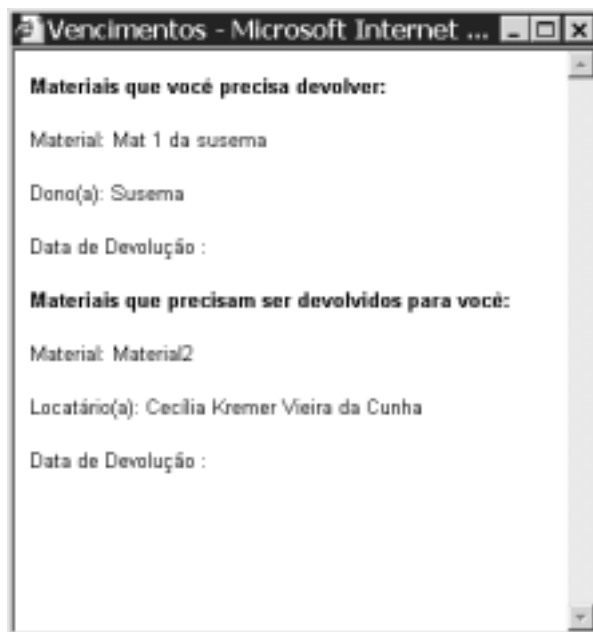


Figura 27. Interface de Visualização de Vencimentos

### Cadastramento de Material

Qualquer usuário pode cadastrar um material de sua propriedade. Caso um usuário não tenha nenhum material de sua propriedade cadastrado ainda, a realização dessa atividade o habilitará a acessar a função de empréstimo, já que o usuário passará ser dono de um material. Veja a rede de interfaces de cadastramento de material:

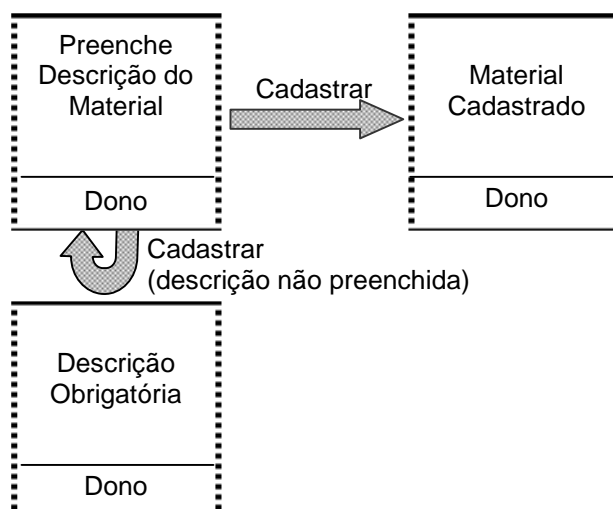


Figura 28. Rede de Interfaces de Cadastramento de Material

E a interface de cadastramento de material:



Figura 29. Interface de Cadastramento de Material

### **Material para Documentação de Extensão**

Além da aplicação, preparou-se também um formulário para documentação das extensões, assim como um conjunto de representações da aplicação original que podiam utilizadas como base para essa documentação. Segue a lista de informações pedidas no formulário entregue aos usuários, junto com explicações sobre as mesmas e sobre as representações oferecidas.

Informações sobre a extensão sugerida

Nome(s) do(s) estendedor(es): Os usuários podiam criar extensões de forma individual ou colaborativa ...

Horário aproximado da construção da extensão: Os usuários informavam o horário inicial e final da construção de cada extensão ...

Descrição breve da extensão: Intenção de design da extensão ....

O uso da extensão pressupõe a participação de outros usuários?  
Para que os usuários pensassem sobre a necessidade de envolver outros usuários ...

Outros usuários poderiam se beneficiar da extensão (mesmo que ela seja de uso individual)? Você a compartilharia? Acha que ela seria bem recebida?  
Para que os usuários pensassem sobre o compartilhamento da extensão com outros usuários ...

**Material para construção da extensão**

O material em anexo apresenta algumas representações do sistema de empréstimos. Ele está sendo fornecido para ajudar a construção da extensão. Manipule-o como desejar: troque folhas de ordem, rabisque, recorte e cole, etc., ou até mesmo deixe-o de lado e use folhas em branco.

Daqui em diante foram fornecidas, em papel, as seguintes representações: Rede de Atividades, Rede de Interfaces, Dumps das Interfaces e o Diagrama de Classes (sem a representação das classes relativas às extensões à aplicação)

Figura 30. Formulário para Documentação de Extensões

## Realização do Teste

O teste foi realizado com um grupo de seis usuários que se conhecem, trabalham juntos e ocasionalmente emprestam materiais uns aos outros. Mais detalhes sobre o perfil do grupo serão descritos na análise de resultados. O local do teste foi um laboratório de computadores,



conhecido pelo grupo de usuários. O grupo foi reunido ao mesmo tempo nesse laboratório e não foi especificado nenhum horário para término do teste.

O objetivo dessa avaliação inicial era que os usuários conseguissem criar e documentar extensões em uma sessão única de trabalho. Quando necessário ao alcance desse objetivo, permitiu-se a comunicação direta com a designer ou, principalmente para os casos de construção colaborativa, entre os usuários. Através da extensão documentada, os usuários realizaram uma comunicação assíncrona com a designer da aplicação. Todas essas comunicações serão discutidas na análise dos resultados.

No início da sessão a avaliadora apresentou uma descrição geral do que seria o trabalho, esclareceu que o objetivo era que se documentassem extensões e em seguida passou para uma explicação breve da aplicação. Os usuários iniciaram então a etapa de utilização da aplicação que teve duração de 20 minutos. Depois a avaliadora explicou as representações em papel e deu início à etapa de criação e documentação de extensões. Os usuários finais utilizaram aproximadamente 1 hora e 10 minutos nesta última etapa.

## **5.2. Análise dos Resultados**

A seguir analisamos os resultados obtidos com essa investigação, tomando como base principal o total aproximado de 45 extensões sugeridas. A aproximação se deve ao fato de que um dos usuários não definiu claramente os limites entre uma extensão e outra. Nesse caso, a avaliadora foi quem estabeleceu os limites entre as extensões, de forma a poder quantificá-las e referenciá-las. Para os outros casos, foram mantidas as unidades de extensão especificadas originalmente. A seguir listamos o número de extensões sugeridas por usuário:

- Usuário 1: 13 extensões (quantificadas pela avaliadora),
- Usuário 2: 11 extensões,
- Usuário 3: 9 extensões,
- Usuário 4: 6 extensões,
- Usuário 5: 4 extensões (uma dessas extensões foi construída em colaboração com o Usuário 6 e contabilizada somente aqui), e
- Usuário 6: 2 extensões.

Cabe destacar que a granularidade de uma extensão variou não só entre usuários diferentes, mas também para um mesmo usuário. Por exemplo, foi possível observar uma extensão constituída de uma seqüência de três interfaces novas, assim como uma outra extensão que modificava apenas uma das interfaces já existentes, sugerindo uma mudança no texto da mensagem de confirmação de empréstimo.

### **Aplicação do Modelo para Análise das Comunicações sobre Extensões**

A seguir mostraremos como aplicamos o modelo proposto para análise das comunicações observadas no teste. A maioria dessas comunicações dizem respeito à comunicação assíncrona entre um estendedor e a designer. Houve também uma comunicação assíncrona cujo emissor era um estendedor e um co-estendedor que construíram colaborativamente uma extensão e o receptor era a designer da aplicação.

Algumas comunicações síncronas foram realizadas durante o teste. Observou-se a realização de 2 processos de diálogo entre sub-grupos de usuários, além do processo de diálogo ocorrido durante a construção colaborativa de extensão. Nenhum desses 3 processos de comunicação síncrona foram analisados, já que não houve registro dos mesmos.

Ainda assim fazemos algumas ponderações sobre a quantidade de comunicações síncronas entre usuários e de construção colaborativa de extensões. Primeiramente, a aplicação permitia que um mesmo usuário alternasse entre os papéis de dono e locatário, uma vez que ele tivesse um material de sua propriedade cadastrado. Por conseguinte, ainda não conseguimos avaliar a situação onde a construção colaborativa seria imprescindível dado que um usuário, por exemplo, não teria acesso a todos objetos necessários para efetivar a extensão. O fato de ser uma aplicação nova para todo o grupo também excluiu a avaliação de outros tipos de comunicação, como a ajuda de um usuário mais experiente a um novato. Além disso, a situação artificial de teste, a presença da avaliadora e o fato de que a disposição das mesas no laboratório colocava a maioria dos usuários de costas uns para os outros podem ter intimidado a realização de comunicações síncronas e construções colaborativas.

Por outro lado, esse resultado inicial constata a existência de construção colaborativa, mesmo quando ela não é imprescindível. O resultado mostra também uma necessidade de reflexão por parte dos estendedores, podendo indicar de fato uma preferência por comunicações assíncronas. Em testes futuros e de duração mais longa pretendemos investigar de forma aprofundada as situações que ainda não foram tratadas no escopo dessa avaliação inicial.

Além das comunicações síncronas entre usuários, foram feitas também algumas consultas à avaliadora/designer durante o teste. A maioria das consultas se referiam a dúvidas e *feedbacks* sobre as representações oferecidas ou o processo de documentação. Um dos usuários questionou a avaliadora sobre se determinada sugestão de extensão constituía de fato uma extensão ou não. Um outro usuário, na documentação de uma extensão, relatou: “não sei se é uma extensão ou uma nova funcionalidade...”. Essas observações apontam para uma necessidade de se esclarecer em sistemas extensíveis não só a possibilidade de criar-se extensões e comunicar-se sobre elas, mas esclarecer também o que são extensões, para que servem, que benefícios trazem, etc. Ou seja, designers de sistemas extensíveis devem prover explicações sobre o conceito de extensões, assim como *affordances* claras às funcionalidades relacionadas às mesmas. Esse resultado da investigação mostra que o conceito de extensão à aplicações nem sempre é claro, até mesmo para usuários que, como veremos, possuem experiência em computação e já haviam sido expostos a esse conceito anteriormente. Essa questão é primordial para o apoio ao desenvolvimento de uma cultura de adaptação e para o tratamento da questão recorrente de por quê usuários finais não utilizam os ambientes de extensão disponíveis.

Vejamos então a análise das comunicações levantadas com base no modelo proposto.

### **Agentes Comunicantes**

Nas extensões documentadas como comunicações assíncronas, temos os seguintes agentes comunicantes:

- Emissores: os estendedores individualmente e, no caso da extensão construída colaborativamente, o estendedor e o co-estendedor;

- Receptor: a designer da aplicação, que também era a avaliadora. Cabe lembrar que os emissores conheciam o receptor e sabiam do seu alto grau de conhecimento sobre a aplicação. Portanto, aplicando as máximas de Grice [Grice'1975], os emissores puderam pressupor que os elementos da aplicação original seriam compreendidos e que a comunicação poderia se concentrar no esclarecimento da extensão sendo proposta, dos novos conteúdos introduzidos e sua relação com os elementos da aplicação original.

As características mais relevantes do perfil dos emissores, que, para atender ao objetivo da avaliação precisavam atuar pelo menos como estendedores, eram conhecimento de programação e familiaridade com a idéia de sistemas extensíveis e construção de interfaces. Assim podemos considerar que os emissores pertenciam à cultura do programador, descrita na seção 2.1. Participaram como emissores quatro alunos de pós-graduação, um pesquisador recém-doutor e um professor de pós-graduação, todos atuantes na área de Interação Humano-Computador.

Embora não tenhamos avaliado ainda comunicações entre estendedores e usuários, as seguintes perguntas do formulário da Figura 30: “o uso da extensão pressupõe a participação de outros usuários?” e “Outros usuários poderiam se beneficiar da extensão (mesmo que ela seja de uso individual)? Você a compartilharia? Acha que ela seria bem recebida?” nos forneceram indicadores sobre o potencial de realização desses tipos de comunicação.

Segundo as respostas dos usuários, havia 4 extensões (dentre as 45) que pressupunham o envolvimento de outros usuários para sua execução, indicando portanto uma necessidade de comunicação com usuários. No entanto, a avaliadora observou que outras extensões teriam algum tipo de impacto em outros usuários e que portanto teriam um potencial para disparar comunicações também. São elas:

- 8 extensões que fazem uso de envio de e-mail para outro(s) usuário(s), e portanto indicam uma participação passiva deste(s). Das 8 extensões, 3 pressupõem uma resposta do destinatário da mensagem, visto que: um e-mail é de pedido de prorrogação de prazo de

devolução e dois são de pedido de devolução de material, sendo que um deles pode envolver cobrança de multa;

- 2 extensões que transferem empréstimo de um locatário para outro;
- 2 extensões que estendem o prazo de devolução de um material;
- 1 extensão que restringe o prazo de devolução de um material;
- 1 extensão que sugere que se mostre a informação de telefone do locatário;
- 1 extensão que sugere a informação de “média de tempo que uma pessoa leva para devolver um livro (principalmente para casos que nunca devolve)”; e
- 1 extensão que apresenta um histórico do locatário.

Esse resultado aponta para a necessidade de um ambiente de extensão a aplicações de grupo que alerte um estendedor para uma análise de impacto do uso da sua extensão sobre o grupo. A utilização da X-DIS como linguagem básica pode ser complementada em um ambiente de extensão com mecanismos que detectem, em extensões, a utilização de construtos (e.g. função de envio de e-mail) que podem causar impacto em outros usuários e assim avisem e incentivem os estendedores a se comunicar com aqueles que serão afetados pela uso da nova extensão.

Em resposta à segunda pergunta, os estendedores julgaram que a maioria das extensões (40 de 45) poderia beneficiar outros usuários e ser compartilhada com eles. Da mesma forma, os estendedores expressaram sua opinião de que essas extensões seriam bem recebidas pelos usuários. Houve a ressalva, por um dos usuários, de que “algumas” extensões, não identificadas individualmente, poderiam não ser bem recebidas. Nesse caso o usuário respondeu: “acho que algumas extensões não seriam bem-recebidas (pois não o foram por mim também)”.

Isso mostra que usuários podem construir extensões e depois querer desistir delas. Para isso, um ambiente de extensão deve permitir a revogação de extensões. Cabe notar que o caso de extensões que envolvem mais de um usuário é crítico e as mesmas não devem poder ser revogadas sem o aceite de todas as partes envolvidas. A X-DIS pode servir de base para

mecanismos que detectem essas extensões, pois poder-se-ia buscar, nas extensões que se quer revogar, os construtos de restrição de acesso aos seus diferentes passos verificando-se se todos eles apontam para o mesmo papel e/ou usuário ou não.

O fato de que a maioria das extensões foi julgada como benéfica a outros usuários, compartilhável e provavelmente bem recebida indica um potencial alto de comunicação entre estendedores e usuários, que pode ser dar de forma unidirecional, como no exemplo de repositórios de extensões a serem compartilhadas [Repenning et al.'1999], ou até mesmo de forma bidirecional e direta, como o exemplo de pessoas que exercem o papel de intérprete definido em [Mackay'1990].

### **Canal de Comunicação**

Como mencionado, o canal de comunicação utilizado para as comunicações assíncronas foi o papel. Esse canal apresenta como principal diferença em relação ao canal computacional uma restrição para a construção de mensagens interativas de fato. Essa restrição foi suplantada com simulações de mensagens interativas onde as mesmas eram expressas através de uma organização em *storyboard*, ou seja, seqüências de *dumps* de telas encadeadas cujo funcionamento mais detalhado era explicado em linguagem natural.

As outras duas características discutidas do canal computacional puderam ser observadas mais diretamente em nossa avaliação: a possibilidade de comunicação através de múltiplos códigos e a possibilidade de realização de comunicações síncronas e assíncronas, onde aqui avaliamos primordialmente as comunicações assíncronas, seguindo a preferência discutida no capítulo 3.

### **Contexto**

No contexto das comunicações assíncronas não havia compartilhamento de tempo nem de espaço. Fazia parte desse contexto: o domínio da aplicação, o *background* dos agentes comunicantes e o ambiente de extensão e comunicação, que aqui se encontravam integrados

através do papel e das representações básicas disponibilizadas para a expressão de mensagens (formulário, *dumps* de tela, etc.).

## **Mensagens**

Analisemos agora o conteúdo das extensões comunicadas. Para isso iremos relembrar o conceito de sistema semiótico específico, definido na seção 2.2 como uma gramática de um sistema de signos particular que descreve um dado campo de fenômenos comunicativos como regidos por um sistema de significação que pode ser estudado a partir de uma visão sintática, semântica ou pragmática.

Como mencionamos no capítulo 3, uma aplicação pode ser vista como um sistema semiótico específico de um domínio e a criação de extensões a ela é influenciada pelos sistemas semióticos que circundam o seu uso. Portanto, há uma tendência de que estendedores queiram integrar elementos (signos) de outros sistemas semióticos que eles já conhecem ao de uma aplicação que é utilizada por eles. Isso pode ser ilustrado pelo exemplo dado por [McDaniel'2001], que observa que pessoas sempre perguntam “O seu sistema pode fazer X?”.

Na fase de preparação para o teste a designer da aplicação fez um mapa dos sistemas semióticos circunvizinhos à essa aplicação, como mostrado na Figura 31. Cada círculo representa um sistema semiótico diferente, mostrando a sua relação com outros sistemas semióticos previstos. Por exemplo, o sistema semiótico do domínio da aplicação envolve tanto signos da linguagem natural quanto signos do ambiente computacional delimitados pelo escopo da aplicação. Dessa forma o mapa destaca influências que poderiam ser exercidas sobre a aplicação e colabora para um exercício de previsão de extensões que poderiam ser sugeridas à aplicação. Signos que já eram observados no sistema semiótico do domínio mas não foram transportados para a aplicação indicavam extensões prováveis, como por exemplo os signos de reserva de material. Como resultado desse exercício, a designer gerou um conjunto de extensões que poderiam vir a ser sugeridas durante o teste.

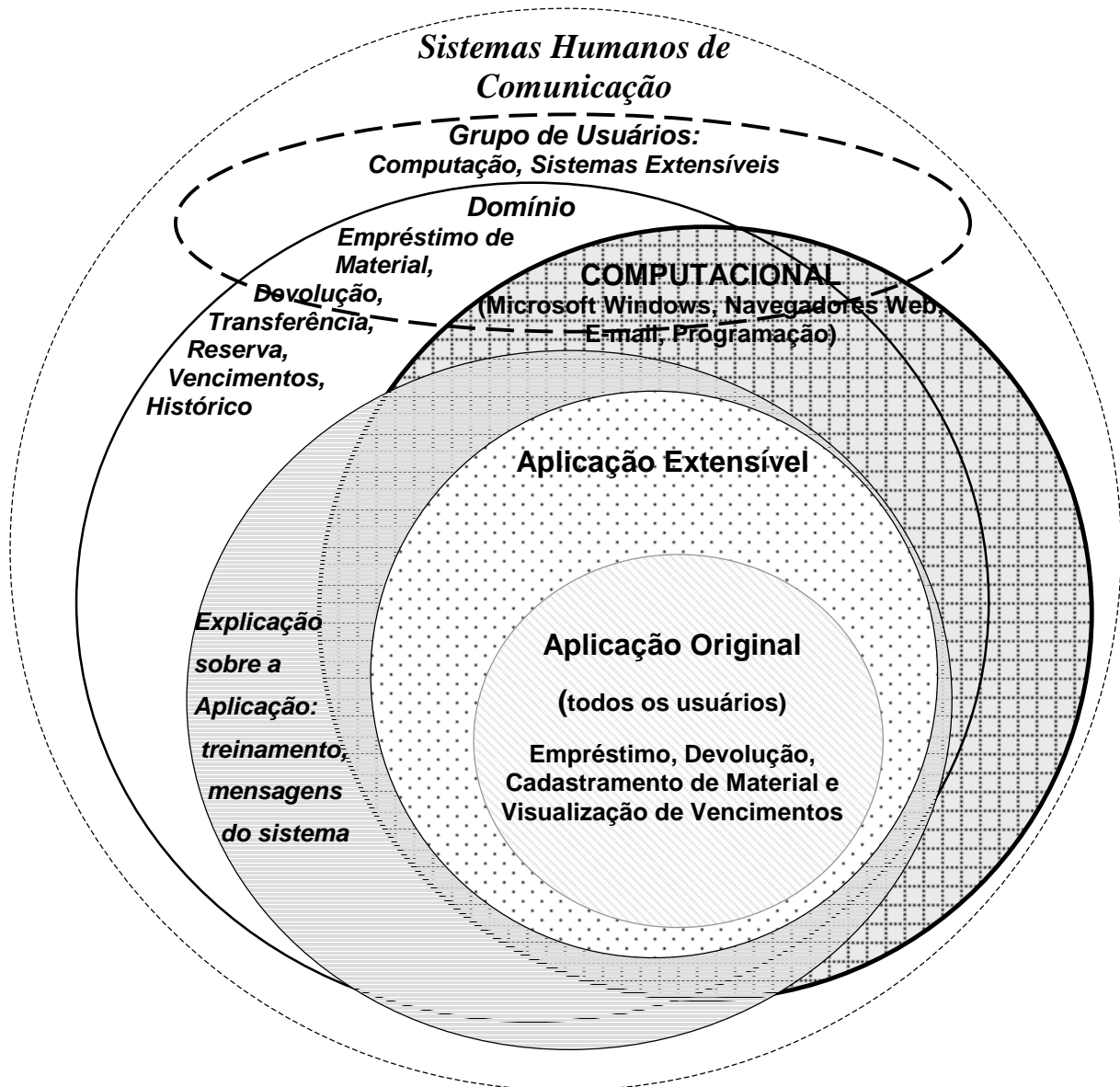


Figura 31. Mapa dos Sistemas Semióticos Previstos

Ao mapear o sistema semiótico do grupo de usuários participantes do teste, a designer criou expectativas sobre signos que seriam utilizados para documentar extensões. Esperava-se, por exemplo, que esse grupo utilizasse conceitos de programação na especificação de extensões, já que eles conhecem linguagens de programação.

Esse exercício nos mostrou também que é possível se fazer uma releitura das pesquisas de modelagem de usuário sob uma perspectiva da construção de extensões, de forma a se prever



em que sentido as linguagens utilizadas pelos diferentes usuários influenciam a criação e especificação de extensões.

Faremos agora uma relação entre as expectativas geradas e os resultados obtidos com a investigação empírica. A seguir mostramos novamente o Mapa dos Sistemas Semióticos e destacamos através de textos sublinhados os elementos que foram sugeridos para adição e/ou alteração na aplicação pelos estendedores.

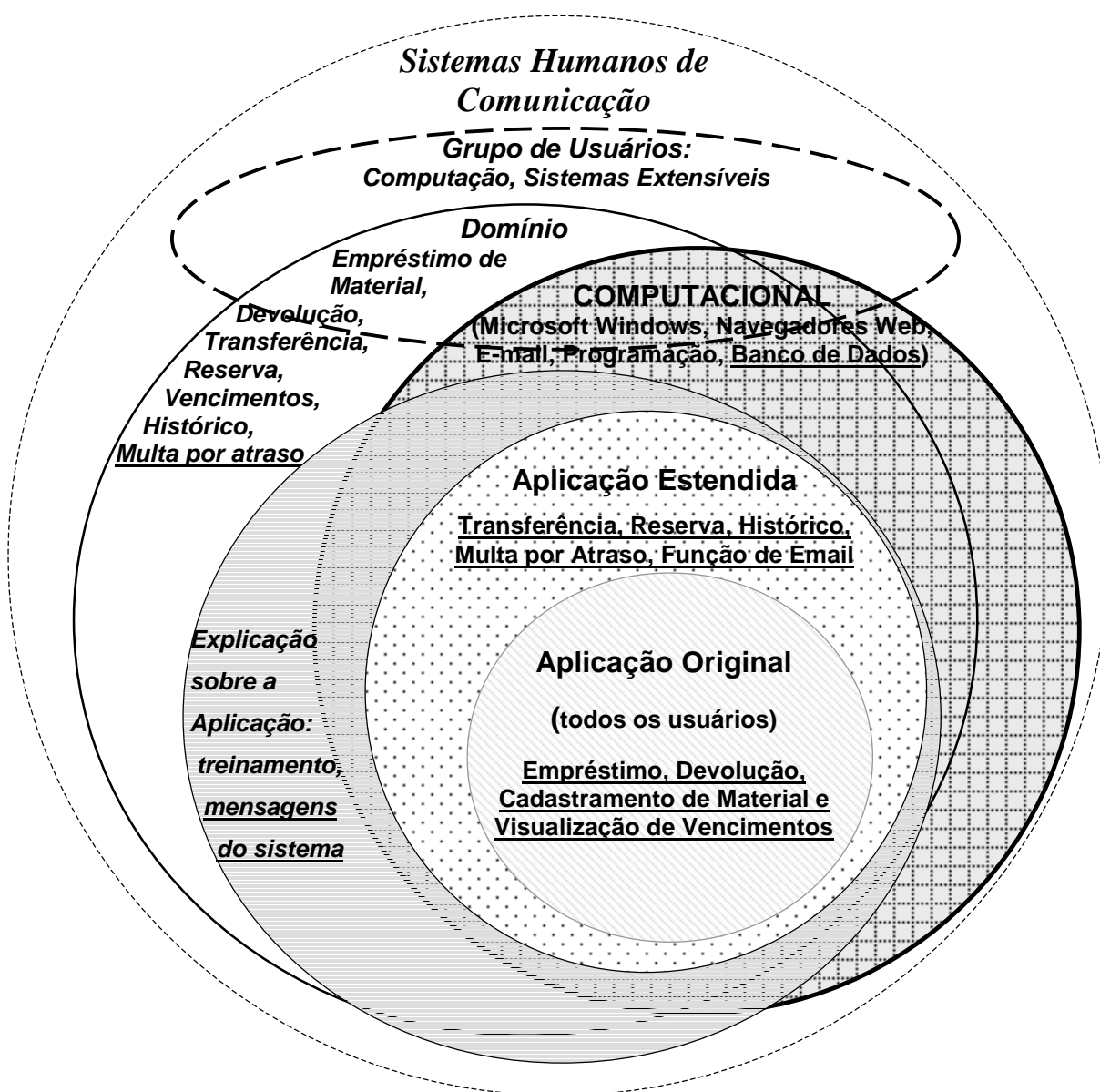


Figura 32. Mapa dos Sistemas Semióticos Obtido após o Teste

Veja-se por exemplo a inclusão do elemento Multa por Atraso, que não tinha sido considerado originalmente no domínio, nem na aplicação. Note-se também a inclusão da função de E-mail, um signo “importado” do domínio computacional. Os usuários expressaram uma série de formas diferentes de apresentação de dados da aplicação, podendo indicar influência de seus conhecimentos sobre linguagens de bancos de dados.

Como dito anteriormente, a construção do mapa inicial ocasionou um exercício de previsão de quais poderiam ser as extensões sugeridas à aplicação. Como resultado, pudemos verificar que, das 45 extensões, 18 delas eram similares às antecipadas pela designer da aplicação, enquanto que 27 delas não haviam sido previstas. Cabe destacar que os exercícios de extensão feitos pela designer tiveram um foco maior em tarefas e representação de informação, o que excluía, por exemplo, extensões para novos *layouts* de interface ou variações a consultas em banco de dados.

Esse resultado indica que o estudo dos sistemas semióticos circundantes pode ser benéfico ao processo de design, já que ajuda ao designer a perceber e organizar o contexto de uma aplicação, enxergando pelo menos em parte o seu potencial de extensão. Isso pode ajudar na construção de sistemas extensíveis mais úteis já que, ao antecipar extensões, um designer pode tomar decisões mais informadas sobre se e como deverá articular os elementos da aplicação na X-DIS para que depois eles possam ser manipulados durante a extensão à aplicação. Por outro lado, o resultado mostra que sistemas extensíveis são mesmo necessários, dado que ainda restou uma quantidade significativa de extensões não previstas.

A seguir apresentamos em mais detalhes quais foram as extensões sugeridas. Primeiramente agrupamos extensões ou partes de extensões que tinham objetivos similares, para que elas fossem analisadas conjuntamente, evitando assim a complexidade de descrever a análise individual de todas as 45 extensões. A documentação com todas as extensões sugeridas originalmente pode ser consultada no Apêndice II.

Ao explicarmos uma extensão indicaremos, através de um índice sobrescrito, sua referência para a matriz de organização do plano de conteúdo e de expressão. Essa matriz será explicada

na seção seguinte com auxílio da Figura 36. O índice utilizado é formado por uma sigla na qual:

- as duas primeiras letras indicam a classificação do conteúdo da extensão, de acordo com a seção 3.1. Assim temos: “II” para Intra-Semiótico Individual, “IN” para Intra-Semiótico Não-Individual e “ES” para Extra-Semiótico Segmentado. Não houve nenhum conteúdo de extensão que se encaixasse na classificação de Extra-Semiótico Não-Segmentado;
- a terceira e a quarta letras da sigla indicam a classificação da expressão da extensão em termos da UIL da aplicação, de acordo com a seção 3.2. Assim temos: “Le” para Expressão Atuante no Léxico, “Si” para Expressão Atuante na Sintaxe e “Se” para Expressão Atuante na Semântica. Não houve nenhuma extensão de Expressão Atuante na Pragmática.

Cada sigla é seguida de um número que indica a seqüência da extensão na célula da matriz. Por exemplo, a primeira menção a uma extensão de conteúdo Intra-Semiótico Individual (II) e Expressão Atuante na Sintaxe (Si) foi indexada de <sup>II</sup>Si<sup>1</sup>.

### **Grupo 1: Instanciação de Workflows**

Foi sugerida a criação de acesso rápido a algumas instâncias de workflows que eram freqüentes, através da criação das funções de “Empréstimo para Fulano”, “Empréstimo para Beltrano”, etc <sup>III</sup>Le<sup>1</sup>. Assim, na interface de preenchimento de informações de empréstimo, o dono não precisava mais preencher quem era o locatário. Esse é o caso típico de automatização de tarefas repetitivas.

### **Grupo 2: Visualização de Dados Existentes**

Este grupo envolve uma série de extensões possíveis. Temos as extensões que modificam apenas o formato de apresentação de dados que já são mostrados no sistema original. Primeiro temos o exemplo mais simples de um usuário que sugeriu que, na devolução, na interface “Escolhe Locatário-Material” (Figura 25), a lista de seleção tivesse o seu *label* alterado de “Escolha o locatário e o material a ser devolvido” para “*Escolha o locatário e o material*

*devolvido*” e o botão “Devolver” fosse modificado para “*Devolveu*”. Essas são as extensões de Mudança de Labels<sup>IIIe2</sup>.

Em outro exemplo de mudança de visualização, sugeriu-se que a interface de vencimentos (Figura 27) apresentasse os mesmos dados, mas da seguinte forma matricial:

**Material que peguei emprestado**

| Material | Dono    | Data de Devolução |
|----------|---------|-------------------|
| x-----x  | x-----x | x-----x           |
| ⋮        | ⋮       | ⋮                 |

**Material que emprestei**

| Material | Locatário | Data de Devolução |
|----------|-----------|-------------------|
| x-----x  | x-----x   | x-----x           |
| ⋮        | ⋮         | ⋮                 |

Sugeriram-se também ordenações diferentes para as listas de opção das interfaces entre outras extensões similares. Referenciaremos essas extensões como Reorganização de Visualizações Existentes<sup>IIISi1</sup>.

Há também as extensões de apresentação mais abrangente, mais restrita ou diferenciada de elementos da aplicação original. Muitas vezes os estendedores querem mostrar mais ou menos informações do que as presentes na interface original. Através da restrição de uma visualização existente<sup>IIISi2</sup> (e.g. quando restringe-se o conteúdo de informação mostrado) faz-se um controle de foco. Por exemplo, um dos usuários sugeriu que se retirasse da visualização de vencimentos (Figura 27) a lista de materiais que ele precisaria devolver. O usuário expressou essa extensão riscando a parte da interface de visualização de vencimentos mensagem que fica sob o título de “Materiais que você precisa devolver:” e escrevendo: “Eu tiraria essa informação, pois acho que ficou desassociada do restante da aplicação.” Aparentemente esse usuário decidiu seguir à risca a perspectiva dominante da aplicação original, que é a do dono do material e/ou gostaria de ter uma interface na qual ele pudesse adotar somente o foco de “cobrador”.

As visualizações diferenciadas simples<sup>II Si3</sup> correspondem a variações de consultas à base de dados da aplicação de forma a responder às diferentes perguntas de cada usuário, mantendo-se ao mesmo tempo uma consistência em relação ao controle de acesso da aplicação. Por exemplo um usuário sugeriu a criação de diferentes visualizações que respondessem às seguintes perguntas: *“P/ quem emprestei determinado material? – Sistema apresenta todos os dados do empréstimo; De quem é esse material que está comigo, etc.”*.

Há os casos onde quer-se fazer uma visualização aumentada simples das informações<sup>II Si4</sup> mostradas em uma interface incluindo-se outras que já foram previstas na aplicação original para aquele usuário. Essas extensões são exemplificadas por aquelas que modificam as mensagens apresentadas nas interfaces da aplicação original, aumentando sua informatividade. Por exemplo, um usuário sugeriu que a mensagem de confirmação de empréstimo fosse modificada de: *“Empréstimo realizado com sucesso.”*, para o seguinte *template*: *“O <material> foi emprestado para <locatário>. [A data prevista de devolução é <data>].”*.

Há também os casos onde quer-se aumentar a visualização das informações<sup>IN Se1</sup> mostradas em uma interface incluindo-se outras que já foram previstas na aplicação original, mas para as quais o estendedor não tem acesso (potencialmente poderia não saber que estão mapeadas na aplicação). Um exemplo de extensão sugerida foi a de que um locatário pudesse ver todos os materiais disponíveis para empréstimo. No entanto, no modelo de grupo original, os locatários só podiam ver os materiais que eles já tinham pego emprestado. Esse é um caso em que a extensão implica na mudança das linguagens subjacentes às da UIL e da EUPL (sem adição de nenhuma informação nova) e, mais ainda, na mudança da mensagem original do designer.

Há também o que chamamos de Visualizações Diferenciadas com Criação de Novo Conteúdo<sup>ESSi1</sup>, que geram respostas a outras dessas perguntas, tais como: *“Quais são os materiais em atraso?; Quais são as devoluções que vencem hoje? E amanhã?, etc.”*. Essas visualizações, embora expressas através de consultas simples à base de dados, implicam na criação de novos conteúdos na UIL: os conceitos de “atraso”, “hoje” e “amanhã” até então não pertenciam ao sistema semiótico da aplicação. Note-se que esse é um exemplo de extensão extra-semiótica que, através da criação de novas visualizações expressáveis, cria

também novas segmentações de conteúdo. Note-se também que esses novos conteúdos já são segmentados em sistemas semióticos externos à aplicação.

### **Grupo 3: Atualização de Dados Existentes**

Esse grupo apresenta três extensões que introduzem novos conteúdos à aplicação:

1. adiantamento ou prorrogação do retorno de um empréstimo através de atualização da data prevista de devolução<sup>ESSi2</sup>;
2. transferência de empréstimo através de atualização de locatário, data de empréstimo e devolução<sup>ESSi3</sup>; e
3. armazenamento de histórico através da re-utilização da data de devolução<sup>IISi5</sup> para indicar a data de uma devolução já realizada, e não a data prevista de devolução. Uma das finalidades de armazenar o histórico era poder traçar o perfil de um locatário ao longo do tempo (e.g. se o locatário costuma permanecer muito tempo com um material).

Todas essas extensões podem ser expressas através de uma manipulação sintática do código da UIL e, quando executadas pelo dono do material, não ferem nenhum controle de acesso. Enquanto as extensões 1 e 2 modificam a UIL sem ferir o funcionamento da aplicação original, a extensão 3 tem o potencial de torná-la inconsistente, já que atribui à expressão “data de devolução” um significado diferente do inicialmente intencionado pela designer. No funcionamento do sistema original, toda menção à “data de devolução” significa uma data de devolução prevista. Caso essa mesma informação passe a ser utilizada para armazenamento de histórico, então teremos que “data de devolução” significará ora “data de devolução prevista” (na interface de empréstimo) e ora “data de devolução ocorrida” (na interface de devolução).

Note-se também que as extensões 1 e 3 podem ser expressas da mesma forma no código da UIL (e.g. uma atualização da informação da data de devolução), no entanto, visam atingir intenções radicalmente diferentes. Essas duas análises levam a duas conclusões importantes:

- que é de fato importante permitir-se a comunicação da intenção de design juntamente com a extensão, e
- que um designer deve ter cautela ao permitir extensões de atualização de dados já que, embora sua expressão seja relativamente simples, a mesma pode modificar a mensagem original do designer de forma que interfere na usabilidade da aplicação. A dificuldade reside no entanto em se conseguir fazer uma identificação clara dessa situação em tempo de expressão da extensão, já que sua construção por si só não é suficiente para indicar a intenção do estendedor.

#### **Grupo 4: Modificação de Papéis**

A maior parte da aplicação original estava voltada para o papel de dono de material e somente a função de visualização de vencimentos podia ser acessada por um usuário que não fosse dono de nenhum material. No entanto, uma vez que o usuário tivesse um material de sua propriedade cadastrado, ele passava a exercer ao mesmo tempo os papéis de dono e de locatário. Enquanto que por um lado este fato favoreceu o levantamento de extensões que visavam o papel de locatário, por outro lado foi observado que dois dos seis estendedores criaram extensões que declinavam seu próprio papel concomitante de locatário, como a extensão mencionada no grupo 1 que “retirava” da aplicação a visualização de materiais a serem devolvidos pelo usuário. Eles assumiram uma postura de cobradores e renegaram as cobranças que poderiam recair sobre eles mesmos. É como se o usuário quisesse restringir ainda mais o sistema semiótico específico da aplicação ao qual tem acesso, por considerar que parte dele é irrelevante.

#### **Relação entre Papel e Atividade**

Nesse grupo, a principal extensão pretendida foi a mudança na relação entre um papel e uma atividade (e/ou interfaces correspondentes). Em geral havia uma transformação do papel passivo de locatário<sup>INSe2</sup>, que só podia ver vencimentos, para um papel ativo no qual o locatário poderia por exemplo registrar um empréstimo feito a ele ou uma devolução de material que estava com ele.

No entanto, normalmente não havia referência explícita a esses papéis. Por exemplo, foi sugerida a extensão de Reserva de Material (que será explicada no grupo 5.1), mas, ao invés de se expressar diretamente que o usuário dessa extensão seria o dono do material, fez-se isso de forma indireta, através da expressão: Reservar “meus materiais”<sup>ESSe4</sup>. O mesmo foi feito para a contraparte desse extensão pelo locatário, ou seja, expressou-se: Reservar “materiais dos outros”<sup>ESSe5</sup>.

Isso pode ser um reflexo do fato que a UIL original não possuía uma forma de expressão para o papel do usuário e que a UIL foi mais utilizada para expressar extensões do que o diagrama de classes (que mostrava esses papéis). Note-se que para uma extensão efetiva, os usuários precisariam de alguma forma para referenciar esses papéis. Restaria levantar portanto se os usuários tinham compreendido de fato o diagrama de classes, indicando que seriam capazes de realizar referências aos papéis quando necessário.

Outro aspecto é que ambos os papéis tinham os mesmos atributos, na verdade os atributos da classe Pessoa (eg. e-mail). A UIL induzia o uso do e-mail para comunicação somente com o locatário, visto que essa informação só era apresentada na tela de dados do locatário. No entanto, algumas extensões propuseram a visualização do e-mail do dono do material<sup>INSi1</sup>. Para construção dessa extensão, poderia interessar a utilização do recurso de analogia, como proposto em [Barbosa’1999, Barbosa&de Souza’1999, Barbosa&de Souza’2000], através do qual um estendedor poderia expressar que “gostaria de ver o e-mail do dono de forma análoga à visualização do e-mail do locatário”.

### **Criação de Novo Papel**

Embora não tenha havido uma criação explícita de novos papéis<sup>ESSe1</sup>, em alguns casos poder-se-ia dizer que essa seria uma consequência da extensão pretendida pelo usuário. Por exemplo, a extensão de Reserva de Material cria um papel de “reservador”, visto que a pessoa que faz a reserva ainda não é um locatário de fato, mas um potencial locatário. Ou seja, possui uma relação diferente com o objeto material em comparação ao locatário de fato, visto que as regras vigentes na relação são diferentes (e.g. não faz sentido mostrar um potencial locatário na visualização de vencimentos).



## Grupo 5: Criação de Novos Elementos

### Grupo 5.1: Criação de Nova Atividade

As novas atividades sugeridas são aquelas apresentadas no grupo 3: prorrogação ou aproximação do retorno do empréstimo através de atualização da data de devolução<sup>ESSi2</sup>; transferência de empréstimo através de atualização de locatário, data de empréstimo e devolução<sup>ESSi3</sup>; e armazenamento de histórico através da re-utilização da data de devolução<sup>IIISi5</sup>; e também a reserva de material<sup>ESSe4 e ESSe5</sup>.

É preciso fazer algumas observações aqui de forma a englobar outras maneiras de se alcançar duas dessas extensões, que não foram mencionadas na descrição do grupo 3. A criação do armazenamento de histórico foi sugerido através da criação de uma nova classe de informação<sup>ESSe2</sup>. A criação da transferência de empréstimo, embora não sugerida dessa forma, foi prevista pela designer como passível de ser alcançada através de uma junção dos workflows das atividades de devolução e empréstimo<sup>ESSi4</sup> (sem a construção de novas interfaces de atualização de dados).

Uma observação que se fez sobre a criação dessas extensões é que elas foram consideradas ora como novas atividades de fato (e.g. foram incluídas na representação da rede de atividades e/ou no menu principal da aplicação) e ora como novas funcionalidades a serem atreladas a outras atividades. Por exemplo, a prorrogação/restrição de prazo de empréstimo, foi considerada uma atividade por um estendedor e, por outro estendedor, foi considerada uma extensão à atividade de visualização de vencimentos. Essa observação mostra uma situação onde, caso houvesse uma comunicação entre esses estendedores, haveria a necessidade de uma negociação de interpretantes sobre essa extensão, já que eles claramente não convergem entre si. Mais ainda, isso indica também um fator a ser considerado na comunicação (inicialmente) unidirecional que se dá a partir de um possível repositório de extensões.

Observamos também algumas extensões que potencialmente seriam construídas colaborativamente. Por exemplo, um locatário que quisesse transferir um empréstimo para outro poderia construir uma extensão na qual ele indicaria o material a ser transferido e para

quem seria feita a transferência e pediria ao dono do material que construísse o restante da extensão que efetivaria a transferência.

### **Grupo 5.2: Criação de Função**

Foi observada a criação da função de envio de email<sup>ESSe3</sup> (a aplicação original mostrava apenas qual era o endereço de e-mail do locatário). Esse exemplo mostra uma possibilidade de forma de comunicação entre estendedores e designer, onde o designer poderia vir a implementar essa nova função e o estendedor a associaria à aplicação original como desejado.

### **Grupo 5.3: Criação de Nova Classe ou Atributo**

Ao longo dessa seção já foram mencionadas algumas classes a serem criadas, como a de Histórico de Empréstimo. Várias outras classes ou atributos foram sugeridos nas extensões e aqui destacaremos alguns deles.

Na criação da Reserva de Material sugerida seria suficiente a criação de um atributo “data de reserva” associado ao material. No entanto, caso se quisesse fazer uma solução mais abrangente, onde um material pudesse ter várias reservas, poder-se-ia pensar na criação de uma classe Reserva. Isso aponta para um benefício de uma comunicação entre estendedores e designers da aplicação, o que permitiria que se pensasse sobre questões que podem não ser levadas em consideração face a um possível imediatismo dos usuários, como por exemplo: situações de contorno e escalabilidade. Outro aspecto que foi notado é que muitas vezes a criação de um atributo era sugerida, mas isso teria implicações em todo o restante do sistema que no entanto eram observadas incompletamente por parte do estendedor.

Para a comunicação de alguns atributos, como a data de reserva, ou classes, poder-se-ia utilizar o recurso de metáfora (“como uma data de empréstimo futura”), embora o estendedor não tenha se expressado dessa forma. É interessante notar que a estrutura do atributo novo e de outro já existente é igual, mudando-se apenas algumas regras atuantes sobre esses atributos. A criação de histórico também se encaixaria na criação metafórica com possível referência a uma estrutura já existente.

Outros atributos sugeridos<sup>ESSe6</sup>, de forma não metafórica, foram: telefone do locatário, média de tempo que um locatário leva para fazer uma devolução, assunto de e-mail, comentário (em empréstimo e devolução), nome de um material, estado de um material (novo, amassado, etc.), quantidade de um determinado material e multa diária. Note-se que todos eles são conteúdos já segmentados fora do escopo da aplicação e cuja inclusão implica em uma modificação na semântica da UIL (pela inclusão de novos elementos).

Algumas extensões apresentaram um alto grau de complexidade e provavelmente exigiriam uma comunicação com o designer ou com algum usuário mais experiente em criação de extensões ou programação. Por exemplo, a criação de média de tempo que um locatário leva para fazer uma devolução foi sugerida como atributo, mas implica em um grau de complexidade maior, com criação de classe de histórico e função de cálculo de média de tempo.

Notou-se também que extensões que poderiam ser construídas pelos estendedores poderiam ser melhoradas a partir de uma comunicação com estendedores mais experientes ou com o designer. Por exemplo, o atributo quantidade (de um mesmo material de um mesmo dono) apresenta duas formas de construção. Poderia ser construído de forma simples, como um contador, através de filtro de banco de dados (COUNT em SQL), ao invés de ser um atributo propriamente dito. No entanto, essa construção é relativamente restrita, já que não permite por exemplo a atualização do total de quantidade de um determinado material. Assim, a atualização deste contador implicaria na criação de funções por parte do designer. Se esse dado fosse implementado como um atributo (agora pelo designer), o sistema como um todo seria impactado e os usuários precisariam estar de acordo com essa mudança. Assim, observamos que o modelo pode apoiar também os processos tradicionais de manutenção de sistemas, oferecendo *insights* sobre as comunicações desses processos, especialmente as que envolvem avaliação de protótipos.

### **Grupo 6: Extensões que Criam Extensões**

Na extensão de empréstimos para locatários mais frequentes (“Empréstimo para Fulano”, “Empréstimo para Beltrano”, etc.), sugeriu-se também a criação de um botão que excluía e

incluía (a partir de um empréstimo corrente) automaticamente do menu principal o acesso a essas funções de empréstimo aos usuários mais frequentes. Outro exemplo sugerido foi um construtor de consultas à base de dados que seriam configuradas e que poderiam ser re-ativadas posteriormente.

A construção dessas extensões caracteriza uma atividade de meta-programação, no sentido em que se dá a construção de um programa que, ao ser executado, gera outros programas. Portanto, ela implicaria na utilização de construtos da X-DIS que gerariam textos novos da própria X-DIS em tempo de execução, textos esses que, logo que gerados, seriam também executados. No estágio atual da representação formal, ainda não há disponibilidade desses construtos, embora nossa intuição seja de que ela poderia ser estendida para acomodá-los.

A seguir apresentamos de forma tabular um resumo dos grupos de extensão observados com seus respectivos exemplos:

|  |  |
|--|--|
| <b>Grupo 1:<br/>Instanciação de<br/>Workflows</b>        | -Empréstimo p/ Fulano, Beltrano, etc. <sup>IIIe1</sup>   |
| <b>Grupo 2:<br/>Visualização de<br/>Dados Existentes</b> | -Mudança de Labels <sup>IIIe2</sup><br>-Reorganização de Visualizações Existentes <sup>IIISi1</sup><br>-Restrição de Visualização Existente <sup>IIISi2</sup><br>-Visualizações Diferenciadas Simples <sup>IIISi3</sup><br>-Visualizações aumentadas (e.g. templates de msgs.) <sup>IIISi4</sup><br>-Aumento na Visualização de Informações (fere controle de acesso) <sup>INSe1</sup><br>-Visualizações Diferenciadas com Criação de Novo Conteúdo <sup>ESSi1</sup> |
| <b>Grupo 3:<br/>Atualização de<br/>Dados Existentes</b>  | -Adiantamento ou Prorrogação da data de devolução <sup>ESSi2</sup><br>-Transferência de Empréstimo pela atualização dos dados <sup>ESSi3</sup><br>-Armazenamento de Histórico c/ a re-utilização da data de devolução <sup>IIISi5</sup>  |
| <b>Grupo 4:<br/>Modificação de<br/>Papéis</b>            | -Transformação do papel do locatário de passivo p/ ativo <sup>INSe2</sup><br>-Reservar “meus materiais”, <sup>ESSe4</sup><br>-Reservar “materiais dos outros”, <sup>ESSe5</sup><br>-Visualização do e-mail do dono do material <sup>INSi1</sup><br>-Criação de Novos Papéis <sup>ESSe1</sup>   |

|   |   |
|---|---|
| <b>Grupo 5:<br/>Criação de<br/>Novos Elementos</b>    | - Adiantamento ou Prorrogação da data de devolução <sup>ESSi2</sup><br>- Transferência de Empréstimo pela atualização dos dados <sup>ESSi3</sup><br>- Armazenamento de Histórico c/ a re-utilização da data de devolução <sup>IISi5</sup><br>- Reservar “meus materiais” <sup>ESSe4</sup><br>- Reservar “materiais dos outros” <sup>ESSe5</sup><br>- Transferência de Empréstimo pela junção de workflows <sup>ESSi4</sup><br>- Armazenamento de histórico através da criação de uma nova classe <sup>ESSe2</sup><br>- Envio de e-mail (criação de novas funções) <sup>ESSe3</sup><br>- Criação de Novos Atributos <sup>ESSe6</sup> |
| <b>Grupo 6:<br/>Extensões que<br/>criam Extensões</b> | - Empréstimo p/ Fulano, Beltrano, etc. <sup>IIIe1</sup><br>- Construtor de Consultas à Base de Dados (engloba os diferentes tipos de visualização abordados)  |

Figura 33. Agrupamento das Extensões Sugeridas

### Códigos de Comunicação

Depois de ver os conteúdos que foram comunicados à designer, iremos analisar como esses conteúdos estavam expressos nos documentos entregues. Primeiramente destacamos que um dos usuários não utilizou nenhuma das representações da aplicação disponibilizadas, mas apenas folhas de papel em branco onde ele desenhou e especificou suas extensões. Portanto, ao mencionarmos os dados quantitativos sobre a utilização das representações estaremos considerando 5 usuários e 39 extensões.

Observamos as seguintes utilizações para as representações entregues:

- ◆ Rede de Atividades: 1 usuário,
- ◆ Rede de Interfaces: 3 usuários,
- ◆ *Dumps* das Interfaces: 5 usuários,
- ◆ Diagrama de Classes: 3 usuários.

Como pode-se perceber, a escolha da representação baseada na UIL foi preponderante entre os usuários especialmente se considerarmos que, das 39 extensões expressas nas representações entregues, 31 delas utilizaram os *dumps* de interface como base de expressão.

Esse resultado dá força às nossas argumentações de que a UIL pode ser vista como código artificial preponderante entre os agentes comunicantes.

As extensões (expressas ou não sobre a representação da UIL) utilizavam também linguagem natural para atingir diversos objetivos: explicar e dar *feedbacks* sobre a representação; descrever e explicar a extensão ou parte dela, o seu funcionamento e regras de funcionamento; justificar a criação da extensão; fazer reflexões e analogias; e documentar comunicação síncrona feita com a avaliadora/designer. Na maioria dos casos, os textos em linguagem natural eram contextualizados na representação de interface. Utilizavam-se apontadores (e.g. setas, símbolos de ‘\*’) que partiam de elementos da interface e chegavam em textos em linguagem natural descritivos da extensão desejada. A Figura 34 mostra de forma integrada duas sugestões de extensão que fazem uso de apontadores e linguagem natural no contexto de uma interface da aplicação original.

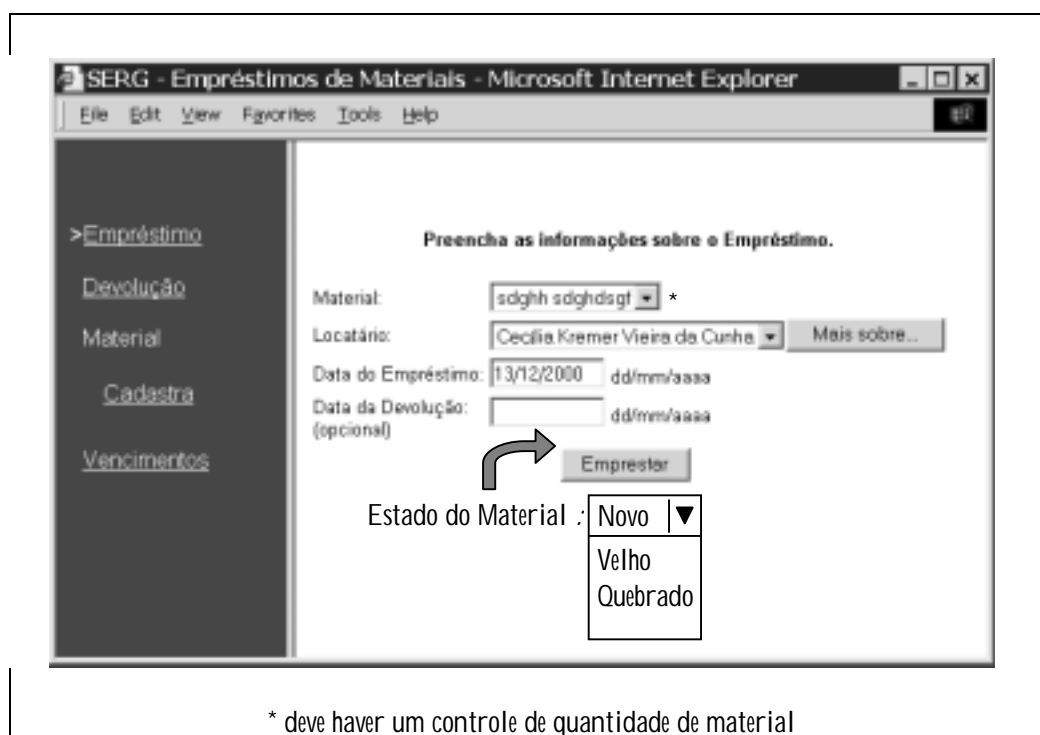


Figura 34. Exemplo de Uso de Apontadores e Linguagem Natural na Sugestão de Extensão

Pôde-se notar também alguns usos de conceitos de programação em algumas extensões. O conceito de variável (e.g. aviso em  $n$  dias), o conceito de contador (e.g. quantidade de

material) e o conceito de valor *default* foram mencionados. Esses conceitos foram mencionados em apenas 4 das 45 extensões. Inicialmente, de acordo com o mapa de sistemas semióticos previstos, esperava-se que a utilização de termos de linguagens de programação fosse relativamente freqüente. No entanto, como também notado em [Silva'2001], os usuários preferiram se expressar em termos mais próximos da linguagem natural, confirmando as pesquisas indicadoras do esforço cognitivo envolvido para expressão nos termos das linguagens de programação convencionais, até mesmo entre aqueles que sabem programar.

### **Elaboração de Mensagens**

Nessa avaliação, o esforço de codificação dos estendedores foi bastante diminuído. A representação em papel, não computável, não apresentou regras gramaticais a serem seguidas, como seria o caso da elaboração de mensagens onde os usuários precisassem aprender e se adequar a um ou mais códigos previstos em um ambiente de extensão e comunicação real. Isso indica a necessidade de testes futuros com sistemas extensíveis implementados e que permitam a construção dessas mensagens formalizadas e uma avaliação mais detalhada do respectivo processo de elaboração de mensagem. Após as discussões relativas à aplicação do modelo, iremos descrever um exercício realizado para uma avaliação preliminar de se e como as extensões sugeridas poderiam ser expressas de acordo com ambientes que implementassem a X-DIS para elaboração de mensagens.

Por outro lado, nossa investigação permitiu a obtenção de dados sobre as representações que os usuários realmente desejariam ter, sem restringi-los aos códigos que poderiam vir a estar disponíveis nos ambientes já implementados. Assim pôde-se fazer uma avaliação crítica do casamento entre as hipóteses de representação propostas pelo modelo, pela linguagem artificial e suas abstrações, e as representações que os usuários de fato gostariam de ter. Outra vantagem obtida foi um número expressivo de extensões comunicadas, que de outra forma exigiria um intervalo de tempo e dedicação maiores dos usuários.

## Compreensão de Mensagens

A maioria das extensões foram compreendidas pela designer da aplicação. Isso se deve a alguns fatores. Primeiro temos que os conteúdos das extensões eram ou conteúdos já previstos no sistema semiótico da aplicação, ou conteúdos já segmentados de outro sistema semiótico (e.g. do domínio). Unindo-se isso à utilização preponderante das representações da UIL, temos que a compreensão das mensagens exigiu na maioria das vezes um processo de abdução hipercodificada, no qual a regra de mapeamento entre expressão e conteúdo é praticamente automática, não apresentando desafios. Houve apenas um caso de abdução hipocodificada, com hipóteses sobre diferentes regras de mapeamento que poderiam ser utilizadas. Nesse caso, como mostrado na Figura 35, o estendedor não esclareceu na visualização de uma lista de reservas de material se o locatário apresentado na interface era um locatário que já estava com o material ou se era o locatário que estava fazendo a reserva. A descoberta da regra adequada dependeria de uma comunicação subsequente da designer com o estendedor, em que se faria o uso da função metalingüística.

|   |  |         |                      |
|---|--|---------|----------------------|
| <div style="border: 1px solid black; padding: 5px; display: inline-block;">meu material</div> | Emprestado até                           |         |                      |
|   | <input type="radio"/> _____              | _____   | _____                |
|   | <input checked="" type="radio"/> x-----x | x-----x | <input type="text"/> |
|   | <input type="radio"/> _____              | _____   | <input type="text"/> |
| <div style="border: 1px solid black; padding: 5px; display: inline-block;">Confirma</div>     |  |         |                      |

Figura 35. Exemplo de Extensão que Ocasinou uma Abdução Hipocodificada



### **Conclusões sobre a Aplicação do Modelo**

Essa primeira aplicação do modelo sobre uma investigação empírica mostrou como ele pode ser utilizado para a caracterizar, organizar e explicar alguns fenômenos de comunicação sobre extensões, permitindo assim uma melhor compreensão das questões relacionadas aos mesmos. A compreensão aponta também para soluções a serem adotadas em aplicações de grupo extensíveis de forma a apoiar a realização de comunicações eficazes, tais como a utilização da aplicação original como base compartilhada entre os agentes comunicantes e a indicação da UIL como código primário.

Pudemos observar então uma primeira utilização do modelo como base de avaliação formativa provedora de referências descritivas para a especificação, implementação, discussão e revisão de projetos de sistemas extensíveis que visam a criação de uma cultura de adaptação não só através de um gradiente de diferentes mecanismos de EUP, mas também através do tratamento explícito dos processos de comunicação que apóiam a formação de cultura.

### **Avaliação Preliminar da X-DIS em relação às Extensões Sugeridas**

De forma a fazer uma avaliação preliminar da linguagem de representação proposta, foi feito um exercício de se e como as extensões sugeridas poderiam ser expressas de acordo com essa linguagem. Nessa seção discutiremos a linguagem de representação tomando como base uma classificação dos grupos de extensões sugeridas segundo a natureza de seus conteúdos e a atuação da sua expressão em relação à linguagem da UIL original, como mostrado na matriz a seguir:

| <b>Conteúdo</b><br><b>Expressão</b>   | <b>Intra-Semiótico Individual</b>  | <b>Intra-Semiótico Não-Individual</b>   | <b>Extra-Semiótico Segmentado</b>   |
|---------------------------------------|--|---|---|
| <b>Expressão Atuante no Léxico</b>    | - Empréstimo p/ Fulano, Beltrano, etc. <sup>IILe1</sup><br>- Mudança de Labels <sup>IILe2</sup>  |   |   |
| <b>Expressão Atuante na Sintaxe</b>   | - Reorganização de Visualizações Existentes <sup>IISi1</sup><br>- Restrição de Visualização Existente <sup>IISi2</sup><br>- Visualizações Diferenciadas Simples <sup>IISi3</sup><br>- Visualizações aumentadas (e.g. templates de msgs.) <sup>IISi4</sup><br>- Armazenamento de Histórico através da re-utilização da data de devolução <sup>IISi5</sup> | - Visualização do e-mail do dono do material <sup>INSi1</sup>   | - Visualizações Diferenciadas com Criação de Novo Conteúdo <sup>ESSi1</sup><br>- Adiantamento ou Prorrogação da data de devolução <sup>ESSi2</sup><br>- Transferência de Empréstimo pela atualização dos dados <sup>ESSi3</sup><br>- Transferência de Empréstimo pela junção de workflows <sup>ESSi4</sup>  |
| <b>Expressão Atuante na Semântica</b> |  | - Aumento na Visualização de Informações (fere controle de acesso) <sup>INSe1</sup><br>- Transformação do papel do locatário de passivo p/ ativo <sup>INSe2</sup> | - Criação de Novos Papéis <sup>ESSe1</sup><br>- Armazenamento de histórico através da criação de uma nova classe <sup>ESSe2</sup><br>- Envio de e-mail (criação de novas funções) <sup>ESSe3</sup><br>- Reservar “meus materiais” <sup>ESSe4</sup><br>- Reservar “materiais dos outros” <sup>ESSe5</sup><br>- Criação de Novos Atributos <sup>ESSe6</sup> |

Figura 36. Matriz Plano de Conteúdo X Plano de Expressão

Nosso exercício mostrou que toda extensão que podia ser totalmente expressa no nível léxico ou sintático da UIL poderia ser expressa através da X-DIS. Isso demonstra o grau de expressividade da linguagem e, observando os respectivos grupos de extensão da matriz (1ª e 2ª linha), indica que ela é útil para a expressão de muitas extensões.

No entanto, nossa investigação constatou também, como já observado na revisão de sistemas de EUP, que extensões criadas nesses níveis podem descaracterizar a UIL original, ou seja, a mensagem original do designer. É o caso por exemplo da re-utilização da data de devolução para fins de histórico, que tornaria a visualização de vencimentos inconsistente. Dado que no nível lingüístico da X-DIS não há diferenciação entre essa extensão e outras perfeitamente úteis e consistentes com a aplicação original (e.g. a transferência de empréstimo), é preciso criar regras e mecanismos inteligentes que atuem no ambiente de extensão de forma a garantir que serão expressos na X-DIS somente textos que não ferem a mensagem original do designer.

Ainda nesse sentido, observamos também que havia extensões independentes que, quando assim utilizadas, não causavam problemas, mas que quando combinadas introduziam inconsistência à aplicação. Por exemplo houve um mesmo estendedor que criou a extensão de prorrogação de prazo de empréstimo e, depois, criou a extensão de reserva de material. Da forma como foram sugeridas, em uma aplicação com essas duas extensões, ocorreria que poder-se-ia estender o prazo de um empréstimo sem observar se havia ou não reserva para aquele material (já que a funcionalidade de reserva não existia quando a de prorrogação foi criada).

Isso demonstra o grau de complexidade atribuído ao mecanismo inteligente, necessário para balancear a tensão entre a necessidade de se construir extensões ao mesmo tempo em que se mantém a aplicação original funcional como base de conhecimento compartilhada entre os agentes comunicantes.

Sobre as extensões que dependem de atuação semântica, é importante notar que elas indicam uma necessidade de extensão nas linguagens de representação subjacentes à X-DIS, como por exemplo: mudanças nos papéis ou restrições do modelo de grupo ou a criação de novas

classes e atributos no modelo de informação. Para tornar essas extensões possíveis, seria preciso integrar em um ambiente de extensão mecanismos de extensão a essas linguagens também, além de mecanismos que possibilitassem integrar essas extensões ao nível da X-DIS, lembrando-se de levar em conta o mecanismo de contínuo semiótico. Outra possibilidade seria apontar para uma comunicação com o designer da aplicação ou com algum usuário mais experiente que pudesse ajudar o estendedor na criação dessa extensão, lembrando que ambas as possibilidades não são exclusivas entre si.

A X-DIS apresenta como vantagem o fato de ser uma linguagem formalizada que pode servir de base para a geração de uma variedade de mecanismos de EUP já propostos. Em um gradiente crescente de poder de expressividade e custo de aprendizado, podemos ver que a X-DIS, em conjunto com derivações das linguagens subjacentes a ela, pode ser utilizada por exemplo para a criação de um gravador de macros, a partir do qual interações com a interface são gravadas como sentenças da X-DIS. Poder-se-ia criar também um formalismo visual que serviria para modificação de macros gravadas ou até mesmo para a criação de extensões totalmente novas, no qual estariam visíveis mais detalhes sobre o funcionamento da extensão e de cada um de seus passos de interação, como proposto em [Cunha et al.'2000a, Cunha et al.'2000b]. A Figura 37 mostra um esquema das linguagens envolvidas nessa solução:

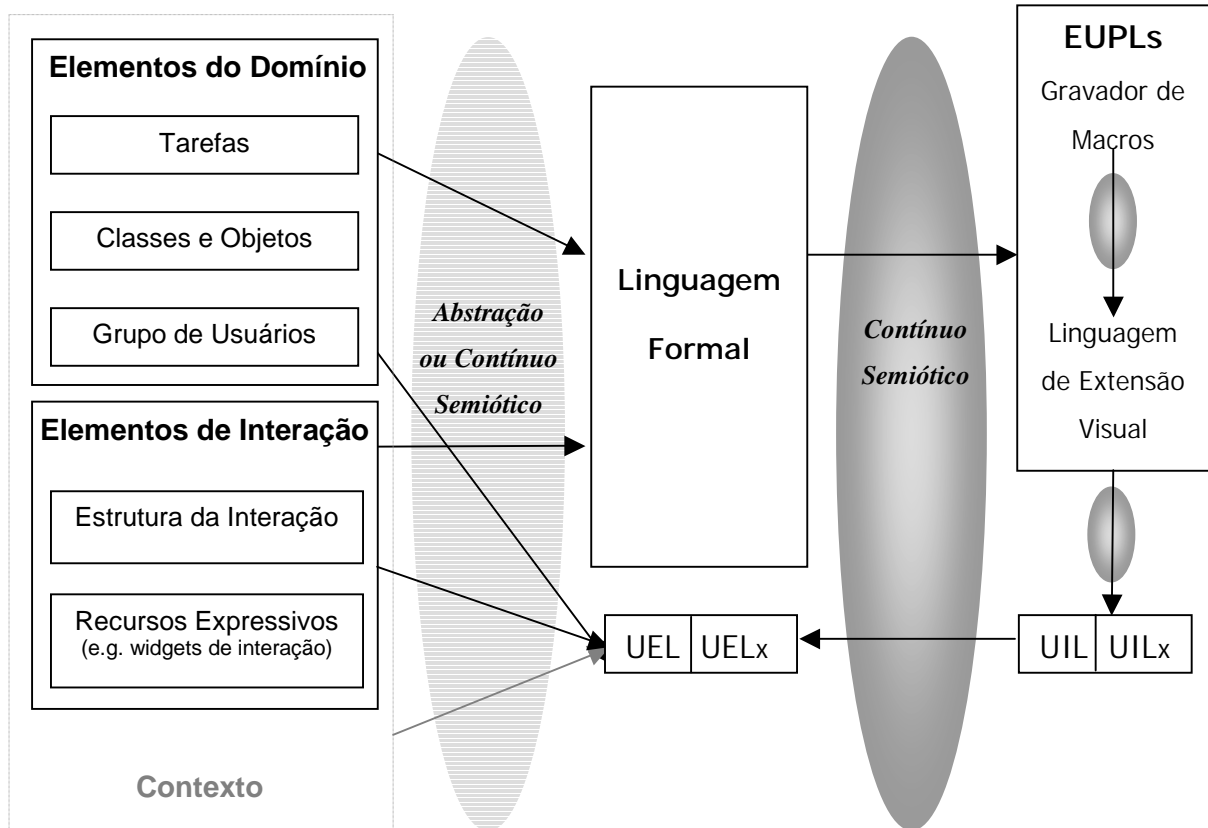


Figura 37. X-DIS e Linguagens Derivadas

O fator importante de ter a X-DIS como ponto de partida para o acesso às linguagens subjacentes à ela é que se mantém como ponto cardeal para a criação e comunicação de extensões a linguagem da interface, ou seja, mantém-se potencialmente o contínuo semiótico entre as extensões construídas e a linguagem de interface, lembrando que a linguagem de interface é a linguagem formal que pode-se pressupor como compartilhada entre os agentes comunicantes. Note-se também que o princípio do contínuo semiótico reforça a necessidade de manter-se uma relação direta entre duas linguagens, facilitando assim um mapeamento que um usuário venha a precisar fazer entre as diferentes soluções providas em um ambiente de extensão, atacando por exemplo o problema mencionado da relação existente entre os gravadores e os editores de macro.

A Figura 37 mostra também que prevê-se que extensões geradas de acordo com a X-DIS tenham representações não só na UIL, mas também na UEL que é a linguagem de explicação

da aplicação e suas extensões. Assim prevemos que possam existir, por exemplo, mecanismos de explicação com geradores automáticos de textos em linguagem natural que transformem os textos codificados de acordo com a X-DIS em textos narrativos, descritivos, explicativos, etc. Esses textos podem visar diferentes tipos de leitores que precisem de níveis variados de detalhe de informação. Por exemplo, usuários potenciais da extensão ou co-estendedores precisam entender não só o efeito de uma extensão mas também o seu comportamento interno. Logo, dizemos que um sistema construído de acordo com a X-DIS é auto-descritível, no sentido em que para explicá-lo não é preciso inferir seu comportamento interno, mas apenas utilizar a própria representação do sistema [Dourish'1997]. Assim, podem-se gerar diferentes descrições confiáveis, já que estas descrições estão diretamente conectadas às representações internas próprias do sistema.

A utilização desses geradores teria uma série de aplicações importantes e úteis. Primeiramente, seria possível que todas as extensões construídas à aplicação tivessem ao menos um texto em linguagem natural associado. Esse texto poderia servir como fonte de trabalho inicial para a elaboração de documentações e comunicações pelos estendedores. Ainda assim, a garantia de ao menos uma documentação automática inicial é de suma importância para a integração dos usuários finais (e.g. trabalhadores) na cultura de adaptação, já que aumentaria as chances de eles conseguirem entender as extensões feitas à aplicação, mesmo sem conhecer a EUPL utilizada para tal.

A representação de acordo com a X-DIS também permitiria a geração de uma variedade de outros tipos de textos. Por exemplo, seria possível gerar sumários sobre todas as referências a uma determinada informação dentro de uma aplicação. Isso poderia ajudar designers e estendedores a prever o impacto da introdução de uma extensão ao sistema. No caso do reuso da data de devolução para fins de histórico, mostrar ao estendedor que essa informação aparece na interface de Visualização de Vencimentos poderia ajudá-lo a perceber que a extensão teria efeitos indesejáveis em outras partes da aplicação. Com a geração de textos poder-se-ia também transformar instâncias armazenadas (e.g. *logs*) de interações dos usuários com o sistema de workflow em narrativas e sumários sobre a realizações passadas de tarefas de domínio que por sua vez poderiam constituir relatórios ou documentos para fins de histórico e auditorias.

Cabe notar que todas essas gerações automáticas e a própria efetivação das extensões dependem de uma codificação de acordo com a X-DIS. No entanto, o que fazer para apoiar as construções e comunicações de extensões que não são totalmente expressáveis na X-DIS, como as que exigem a criação de novos elementos semânticos na linguagem? Para isso sugerimos que se ofereçam aos estendedores mecanismos de denotação que permitam especificar parcialmente ou ao menos contextualizar a extensão proposta em relação à aplicação original. Imagine-se por exemplo que um estendedor queira criar uma extensão à aplicação de Empréstimo de Materiais para permitir o envio de e-mail a partir da interface de Visualização de Vencimentos, cuja versão original foi apresentada na Figura 27. O estendedor passa então para um editor visual de extensão e adiciona um botão de envio de e-mail à interface como mostrado a seguir:

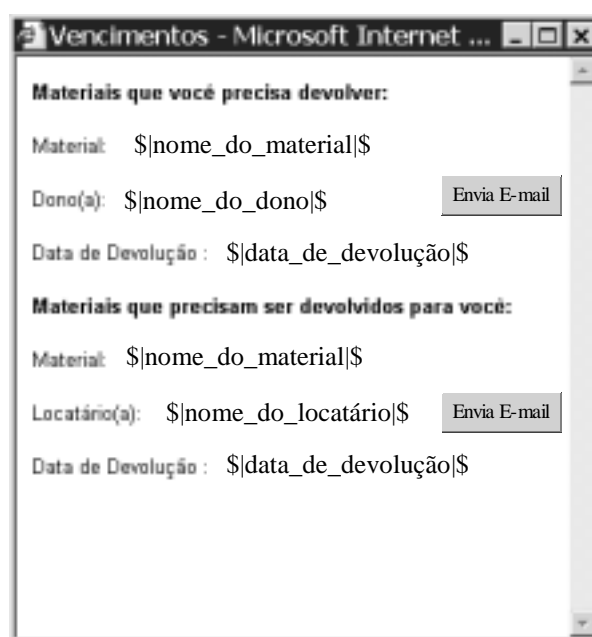


Figura 38. Edição Visual da Interface de Visualização de Vencimentos

O próximo passo seria associar esse botão à função de e-mail. No entanto, a aplicação original não possui essa função e isso impede a implementação completa da extensão. Assim o estendedor precisaria se comunicar com o designer da aplicação, ou com um estendedor mais experiente (e.g. o intérprete), para pedir que essa função fosse integrada ao modelo da aplicação original. Nossa sugestão é a de que a implementação parcial mostrada na Figura 38

fizesse parte do conteúdo dessa comunicação, representando uma parte da especificação da extensão desejada.

O estendedor poderia querer ir mais além da especificação e descrever de forma contextualizada detalhes da função de e-mail específica que ele deseja. Ele poderia querer dizer por exemplo que cada um dos botões de envio de e-mail ativam funções diferentes. Nossa sugestão é a de que, além da especificação parcial já descrita, ele pudesse também referenciar elementos da aplicação e/ou da extensão e adicionar textos explicativos correspondentes, como mostrado na Figura 39:

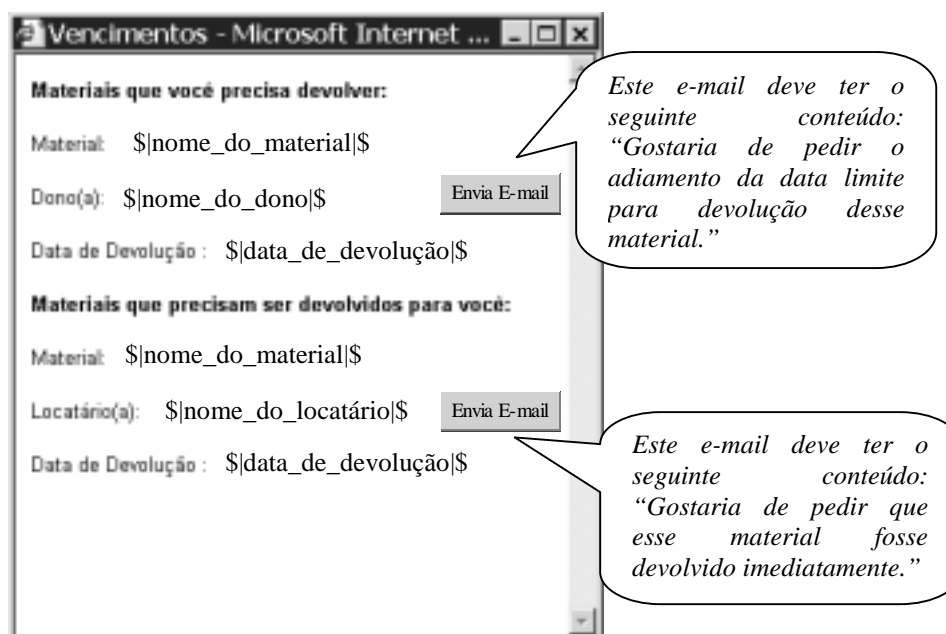


Figura 39. Anotação sobre a Edição da Interface de Visualização de Vencimentos

Assim mostramos como textos parciais ou integrais da linguagem X-DIS podem funcionar como referenciais lingüísticos integrados que articulam diferentes dimensões de uma aplicação (elementos de interface, de informação, etc.) e possibilitam representações manipuláveis de cada dimensão, prevendo-se um ambiente que permita um gradiente denotacional de comunicação de extensões que varia desde implementações formais completas, a especificações semi-formais e a descrições em linguagem natural que utilizam essas representações como base de referência.



Ao utilizar a X-DIS nesse sentido, poder-se-ia dizer que chegamos mais perto do ideal de [MacLean et al.'1990] de tornar os mecanismos de adaptação mais acessíveis e tornar mais linear e gradativo o esforço de aprendizado exigido de uma pessoa na medida em que ela obtém poder de adaptação de um sistema. Como consequência, teríamos uma diminuição na discrepância existente entre as culturas dos trabalhadores, dos exploradores, e dos programadores. Unindo-se essa solução à disponibilidade de um ambiente de comunicação integrado, podemos dizer que chegamos mais perto de viabilizar o uso difundido das soluções de EUP e formar uma cultura integrada de adaptação.

## 6. Discussão

Neste capítulo fazemos uma apreciação de nosso trabalho a partir de seus correlatos, ressaltando seu diferencial e mostrando aspectos que ainda devem ser tratados. Iniciamos com trabalhos que tratam das práticas colaborativas de EUP e depois consideramos ferramentas de EUP e de construção de sistemas adaptáveis, tratando também de alguns trabalhos de sistemas de workflow flexíveis.

O corpo de pesquisa sobre as práticas colaborativas de EUP é constituído principalmente por investigações empíricas que evidenciam e analisam a existência dessas práticas. Nosso trabalho toma essas pesquisas como motivação e, como contribuição, oferece uma abordagem teórica dos fenômenos comunicativos relacionados às práticas observadas. A abordagem resulta em um modelo que organiza aspectos estruturais e comportamentais relevantes e distintivos desses fenômenos e, assim, serve como plataforma para futuros métodos e ferramentas que pretendam apoiar a atividade de extensões a aplicações prevendo as respectivas práticas colaborativas.

Trabalhos como os de [Clement'1990, Mackay'1990, MacLean et al.'1990 e Nardi'1993] mostram que processos colaborativos que visam a adaptação de um sistema se desenvolvem a partir da introdução de um sistema adaptável em um ambiente social. Esses processos envolvem comunicações sobre adaptações. [MacLean et al.'1990] mostram que a dificuldade relacionada a essas comunicações está atrelada ao fato de que os agentes comunicantes normalmente pertencem a culturas distintas onde há uma variação de conhecimento sobre computação.

Nosso trabalho complementa essa afirmação a partir de uma perspectiva semiótica e mostra que a dificuldade de comunicação também é influenciada pelo fato de que extensões introduzem novos significados aos discursos de uma aplicação original e que não se pode pressupor que esses novos significados já sejam compartilhados entre os agentes comunicantes. Como em uma comunicação é preferencial a escolha de signos conhecidos à

criação ou aprendizado de novos, vimos também que é interessante que, sempre que possível, a extensão seja expressa em termos do código da UIL valendo-se do conhecimento compartilhado entre os agentes comunicantes. Assim minimiza-se o esforço do emissor à codificação e ao destaque do novo significado que está sendo introduzido pela extensão e que não faz parte da UIL original. A contrapartida do lado do receptor quanto à decodificação e compreensão da mensagem também é válida.

O trabalho de [MacLean et al.'1990] propõe uma aproximação das diferentes culturas através do oferecimento de um sistema de construção de adaptações que acomode um gradiente de técnicas direcionadas aos estendedores das diferentes culturas. Esses estendedores são apoiados também pela figura do faz-tudo (ou intérprete), que consegue atuar nas diversas culturas.

Nosso trabalho oferece uma linguagem de representação que pode ser utilizada como base para um sistema de adaptação como o de [MacLean et al.'1990], no sentido em que permite a utilização de diferentes técnicas de construção de extensões que abranjam as diferentes culturas. Assim como em [MacLean et al.'1990], nossa linguagem garante que extensões sejam expressas no nível da interface. No entanto, através da aplicação do princípio do Contínuo Semiótico e da inclusão do ciclo mínimo de interação na estrutura da linguagem, garantimos não apenas uma aparência da extensão que permita a sua chamada na interface, mas também uma resposta ao usuário.

Adicionalmente, nossa linguagem sempre permite a visão de uma aplicação e suas extensões prontas segundo uma perspectiva de seus passos interativos. Já em [MacLean et al.'1990], a visão de uma extensão se dá a partir de parâmetros especializados configuráveis ou do código LISP associado ao botão de interface que invoca a extensão. Ao garantirmos que o entendimento de uma extensão pode partir de sua visão interativa, portanto mais próxima da linguagem familiar da UIL, aproveitamos a base de conhecimento compartilhada entre as diferentes culturas para prover uma maior qualidade no apoio aos processos comunicativos que se dão entre elas. Note-se também que a codificação da extensão de acordo com a X-DIS permite uma geração (automática) de outras formas de expressão da extensão, como por

exemplo documentos ou sumários em linguagem pseudo-natural, como discutido no final da seção 5.2.

Nosso trabalho se diferencia também por atuar no escopo definido por uma aplicação. No trabalho de [MacLean et al.'1990] usuários compartilhavam botões através de e-mail que depois eram associados à aplicação. Já nosso modelo aponta para comunicações no contexto da aplicação, permitindo assim um uso menos custoso da função referencial e portanto melhoria nos recursos disponíveis para as comunicações. Cabe lembrar que segundo [Jakobson'1960], a função referencial é bastante utilizada em processos de comunicação. Poder-se-ia argumentar que a adição de um ambiente de comunicação sobrecarregaria uma aplicação, seu custo de desenvolvimento e aprendizado. No entanto, a situação atual sobrecarrega os agentes comunicantes que precisam fazer um esforço maior de comunicação já que não podem fazer referências diretas ao contexto da aplicação. Uma comunicação contextualizada evita também problemas de usuários que queiram associar uma extensão à uma outra aplicação, diferente da aplicação para qual a extensão foi criada originalmente.

Como consideração final sobre o trabalho de [MacLean et al.'1990], destacamos a afirmação de que a existência de uma cultura integrada de adaptação passa pelo tratamento da questão de adaptação como um esforço comunitário e é nesse sentido, e com uma perspectiva comunicativa, que atua nosso trabalho .

O trabalho de [Mackay'1990] mostra uma avaliação do compartilhamento de extensões realizadas para adaptar o uso do ambiente Unix. As extensões eram de diferentes naturezas, tais como a escolha de um editor de texto como preferencial ou a definição da aparência ou forma preferencial de interação com uma aplicação. As extensões podiam ter efeito em uma ou mais aplicações. É preciso fazer uma ressalva de que esse trabalho não avaliou o custo de aprendizado associado aos diferentes tipos de extensão.

Uma das conclusões de [Mackay'1990] é a de que é necessário que softwares extensíveis apoiem explicitamente o compartilhamento de extensões. As formas preferenciais de compartilhamento de extensões variam de acordo com o perfil do estendedor. Estendedores com perfil altamente técnico (análogo ao conceito de programador) preferem compartilhar

extensões por meio de mecanismos de *broadcasting* [Mackay'1990]. Embora eles desenvolvam extensões a partir de experimentações com o software, sem intenção de atender às reais necessidades dos usuários, os usuários as adotam como forma de evitar o custo de aprendizado de como construir extensões. Estendedores com perfil técnico e conhecedores do domínio da aplicação (e.g. intérpretes) desenvolvem extensões com base nas necessidades dos usuários e preferem compartilhar suas extensões através de uma comunicação direta com eles [Mackay'1990].

A investigação de [Mackay'1990] mostra também as principais estratégias para realização de extensões. Nota-se que, exceto para o caso dos estendedores com perfil altamente técnico, a principal estratégia para realizar extensão era tentar obter uma já pronta, através de consulta a outras pessoas. A segunda estratégia era copiar uma extensão de outra pessoa e alterá-la de forma a atender às necessidades pessoais. As outras estratégias, em ordem decrescente de uso eram: consulta a manuais, documentação e códigos fontes e escrita de um código totalmente novo.

Essas observações atestam a necessidade de softwares extensíveis que apoiem comunicações sobre extensões, tanto de forma unidirecional quanto bidirecional e direta. Por conseguinte, é necessário apoiar também os designers dessas aplicações, para que eles tratem diretamente e apropriadamente dessa demanda por comunicação.

Nosso trabalho atua nesse sentido, através da proposta de um modelo teórico e de uma linguagem de representação dos discursos das comunicações sobre extensões a uma aplicação. A linguagem adota primariamente uma perspectiva da interação da aplicação, valorizando-a como base de conhecimento compartilhada entre os agentes comunicantes das diferentes culturas. Ela atua como integradora dos diferentes elementos que compõem um Sistema Orientado à Tarefa, representando seus passos interativos de acordo com o formalismo de ATN. Mostramos, tanto nesse trabalho quanto em [Cunha et al.'2000a, Cunha et al.'2000b], exemplos de possíveis utilizações dessa linguagem em sistemas de workflow. Assim tratamos não só de extensões para uso pessoal, mas extensões de construção e uso intrinsecamente compartilhados. Mencionamos também alguns mecanismos de EUP que poderiam viabilizar o uso da X-DIS por usuários finais.

Dentre os mecanismos de EUP que vislumbramos para a X-DIS, o que mais se aproxima de nossa proposta é o da ferramenta ACE (*Application Construction Environment*) [Zarmer&Chew'1992, Zarmer et al.'1992, Johnson et al.'1993], descrita na seção 2.1. A ferramenta foi construída com o objetivo de atender aos diferentes gradientes de extensão possíveis, visando desde o usuário final até o programador profissional. Propõe-se que os usuários finais construam a maior parte da aplicação, tomando como base ambientes extensíveis preparados por programadores. A construção da aplicação se dá basicamente a partir do acoplamento de objetos da biblioteca de classes a elementos de um formalismo visual, e da descrição de comandos na linguagem de extensão.

Nossa proposta se diferencia da ACE sob vários aspectos. Primeiro temos que o objetivo primordial da ACE é possibilitar que usuários construam aplicações simples apenas para eles mesmos [Zarmer et al.'1992]. Os autores dizem que quando uma aplicação ultrapassa o escopo de uso pessoal, a noção de compartilhamento passa a ser importante e a qualidade do design dessa aplicação é crítico. Eles argumentam que essas aplicações precisam de um apoio de usuários mais experientes, para os casos de extensões que valham para todo o grupo de usuários, de forma a manter a qualidade e padronização das novas aplicações.

Nesse trabalho propomos que uma aplicação inicial pronta seja oferecida a um grupo por um designer, que portanto será responsável pela qualidade do design original. As extensões são construídas sobre uma aplicação pronta, ou seja, o ponto de partida para os estendedores já oferece interações prontas que podem servir como exemplos de design com qualidade. Cabe destacar que essa proposta leva em consideração a investigação de [Mackay'1990] que mostra que essa é a segunda principal estratégia de construção de extensões. Já na proposta da ACE não há, a princípio, a disponibilidade desses exemplos, embora a escolha de um formalismo visual específico (e.g. tabelas, grafos, etc.) induza um tipo de organização das informações da aplicação.

Em trabalhos futuros pretendemos estudar mais profundamente a questão da qualidade do design das extensões produzidas e oferecidas a outros usuários. Nosso ponto de vista é o de que, ainda que não se possa garantir uma qualidade no design das extensões, a possibilidade de construção e efetivação das mesmas deve ser mantida. Note-se que qualquer extensão,

independentemente do seu grau de qualidade, não só dá chance ao usuário de adaptar um sistema à sua realidade como também atua como elemento de comunicação e mediação de conhecimento entre agentes comunicantes. Reconhecemos que extensões de baixa qualidade podem impactar negativamente o trabalho dos usuários, no entanto, a impossibilidade de construção de extensões também produz impacto negativo pois implica em uma adaptação potencialmente indesejada do usuário (e não do sistema). Além disso, a impossibilidade de construção de extensões também impediria que os processos de construção e compartilhamento atuassem como aproximadores das diferentes culturas envolvidas no uso de um software. Portanto, em nossos trabalhos futuros pretendemos estudar mecanismos que ajudem a lidar a questão da qualidade do design das extensões, não só através do apoio ao design com qualidade como também através de mecanismos que permitam identificar as extensões de maior ou menor aceitação, tal como os mecanismos de seleção natural (embora o critério de aceitação não aponte necessariamente para uma alta ou baixa qualidade de design).

Além disso, [Zarmer et al.'1992] partem do princípio de que extensões a uma aplicação de grupo tenham sempre efeito no âmbito de todo o grupo. Não é essa a nossa proposta para um sistema funcional que venha a aplicar nosso modelo e linguagem de representação. Propomos que toda extensão à aplicação tenha efeito inicial somente no escopo do estendedor e, caso haja, do co-estendedor, que a construiu. A construção colaborativa e/ou a adoção de uma extensão de um estendedor por um usuário é possível, mas só ocorrerá quando ambos consentirem. Assim, podemos dizer que um usuário tem, a princípio, poder de decisão sobre aceitar ou não um design proposto por um estendedor. Logo, não é possível a um estendedor construir uma extensão que impacte o trabalho de outro usuário sem o seu consentimento. A manutenção de uma aplicação que impacte todo o grupo de usuários ocorre em duas situações distintas: através de uma adoção generalizada de uma extensão compartilhada a todos os usuários por seu estendedor (e.g. a disponibilização da mesma em um repositório), o que normalmente decorrerá de um movimento espontâneo do grupo; e através de uma modificação à aplicação original construída pelo designer ou por estendedores mais experientes autorizados para tal.

No primeiro caso de manutenção, assim como nas extensões que não afetam o grupo todo, a UIL original não pode ser alterada e então a base de conhecimento compartilhada é mantida.

Já o segundo caso de manutenção pode implicar em uma mudança dessa UIL original. Dizemos então que esse caso força a criação de uma nova versão da aplicação, o que implica em uma re-avaliação de todas as extensões construídas à aplicação da versão corrente, ou até mesmo a criação de uma outra aplicação. A investigação de [Mackay'1990] mostra a grande resistência de usuários que utilizam um sistema estendido a esse tipo de mudança que, portanto, deve ser feita com cautela, observando-se aspectos de compatibilidade com a versão anterior.

É importante notar que a ACE é de fato um mecanismo de extensão de aplicação, enquanto que nosso trabalho oferece uma linguagem formal que pode servir de base para a derivação de diferentes mecanismos de extensão, como explicado no capítulo de avaliação. Cabe destacar que ela também poderia servir de base para a geração de textos em linguagem pseudo-natural que descrevessem novos workflows, instâncias de workflows, etc. como explicado no final da seção 5.2. Assim, uma diferença essencial entre alguns mecanismos de EUP e o nosso trabalho reside no fato de que visamos apoiar de forma intrínseca os processos de comunicação sobre extensões.

Como dito anteriormente, a ferramenta ACE visa a construção de aplicações mono-usuário. Vejamos agora a ferramenta Oval que visa permitir a construção de um amplo espectro de aplicações para trabalho cooperativo [Malone et al.'1995]. O objetivo é oferecer aos usuários uma ferramenta radicalmente adaptável que pode ser utilizada para construção, por exemplo, de sistemas de workflow. Na Oval a criação de aplicações e extensões são realizadas a partir da utilização dos seguintes blocos primitivos de construção de extensões: Objetos, Visões, Agentes e *Links*. A proposta visa atender a diferentes grupos de estendedores, abrangendo o gradiente que vai de usuários finais, que fariam normalmente extensões a aplicações prontas, a programadores, que construiriam aplicações novas a partir dos blocos primitivos. Prevê-se também que as aplicações e extensões construídas possam ser compartilhadas através de e-mail, de bancos de dados ou de arquivos.

Os autores afirmam que não atacam a questão de como gerenciar o processo de extensão de aplicações para grupos de usuários. Eles explicam a necessidade de que o grupo de usuários tenha um entendimento compartilhado sobre a aplicação e descrevem que ferramentas como a



Oval podem colaborar para chegar-se a esse entendimento através: (1) da eliminação de programadores como intermediários; (2) da facilidade para o desenvolvimento e evolução rápida de uma aplicação e (3) da ajuda na gerência da comunicação e tomada de decisão.

Nosso trabalho complementa o de [Malone et al.'1995] no sentido em que atua justamente na questão do uso e desenvolvimento de conhecimento compartilhado. Podemos dizer que nossa proposta contribui para o item (1) já que, como dito no capítulo de avaliação, a idéia é que, a partir da X-DIS, derivem-se diferentes mecanismos de extensão que atuem em diferentes gradientes de extensão [MacLean et al.'1990, Nardi'1993], desde o estágio que envolve um baixo custo de aprendizado e baixo poder de expressão de extensão (e.g. a gravação de macro) até o estágio que envolve um alto custo de aprendizado e alto poder de expressão de extensão (e.g. uma linguagem de programação completa que permita inclusive a extensão dos elementos que são integrados pela X-DIS).

Nosso trabalho atua também no item (2), no entanto o diferencial está no fato de que os blocos primitivos de nossa proposta, ou seja, os referenciais básicos para o grupo de usuários, estão na mensagem original do designer como transmitida através da UIL. Assim, fazemos a ressalva de que a construção de extensões não deve possibilitar a destruição da UIL original. Caso contrário, a base de conhecimento compartilhada entre os agentes comunicantes estaria sendo constantemente destruída e re-construída e esse processo tornaria as comunicações ineficazes.

Em relação ao item (3), podemos dizer que é interessante que um ambiente de comunicação que adote nossa proposta seja complementado com trabalhos de gerência de comunicação, já que nosso trabalho focalizou no problema de introdução e negociação de significados, e não na formalização da expressão da intenção da comunicação ou na organização e recuperação dessas comunicações. Como alguns exemplos de trabalhos que atuam nessa linha citamos o projeto Coordinator [Winograd'1987], ACCORD [Laufer&Fuks'1995] ou a extensão que está sendo proposta ao modelo de grupo adotado em nosso exemplo [Barbosa et al.'2001]. O mesmo ocorre para os processos de tomada de decisão, onde aqui apontamos para pesquisas como a da ferramenta gIBIS [Conklin&Begeman'1988].

Mostramos, tanto nesse trabalho quanto em [Cunha et al.'2000a, Cunha et al.'2000b], exemplos de possíveis aplicações de nossa proposta em sistemas de workflow. Assim é preciso discuti-la à luz de pesquisas existentes sobre sistemas flexíveis de workflow.

O trabalho de [Glance et. al.'1996] se aproxima do nosso no sentido em que propõe o uso de uma gramática como formalismo para representar e processar workflows flexíveis. A gramática representa de forma integrada as abordagens a sistemas de workflow orientadas a atividade e orientadas a documentos [Abbott&Sarin'1994]. Nossa linguagem, em conjunto com as linguagens subjacentes a ela, especialmente o modelo de tarefas e de informação, também permite essa integração. Por exemplo, ao realizar tarefas durante a interação com uma aplicação, um usuário pode vir a preencher documentos (e.g. registros de empréstimos) e como consequência, ativar e/ou desativar um conjunto de tarefas (e.g. a devolução do material emprestado).

Mas diferentemente de [Glance et. al.'1996], nossa proposta visa prover uma representação que possa ser entendida, manipulada e comunicada entre usuários finais e por isso adota a perspectiva de interação. Dessa forma, é possível a um usuário utilizar a linguagem da interface, já conhecida, como ponto de partida para realizar extensões à aplicação e atuar no nível das representações propostas por [Glance et. al.'1996].

Os exemplos de possíveis aplicações de nossa linguagem trataram de processos nos quais documentos ou partes de documentos que eram atualizados por tarefas diferentes e de forma independente (mapeamento 1-1 entre documentos ou partes de documentos e tarefas). Esses processos foram chamados de *decomposable document processes* por [Glance et. al.'1996]. Os exemplos trataram também de processos onde documentos, ou suas partes, eram atualizados de forma seqüencialmente dependente (mapeamento onde documentos são atualizados por uma seqüência de diversas tarefas). Esse é um dos tipos de processos nomeados como *non-decomposable document processes* pelos autores. Ainda não exploramos os casos onde um documento poderia ser manipulado por mais de uma tarefa de forma simultânea (e.g. duas pessoas editando um mesmo texto ao mesmo tempo). Nossa intuição é de que o tratamento desses casos implicará em mudanças nas representações subjacentes à X-

DIS, mas que estas não causarão impacto sobre a estrutura geral atual da X-DIS. Esse é um ponto a ser explorado e verificado em trabalhos futuros.

Outra proposta para desenvolvimento de sistemas de workflow flexíveis é a FreeFlow [Dourish et al.'1996]. Nessa proposta um usuário pode escolher, em tempo de execução, entre diferentes estratégias para atingir um determinado objetivo. Isso é permitido a partir do uso das técnicas de Inteligência Artificial de encadeamento para frente ou para trás em conjunção com um formalismo de modelagem de processos onde cada tarefa está associada a um conjunto de restrições, de forma análoga à programação baseada em restrições, ou *constraint-based programming*. Há restrições que podem ser quebradas durante a execução de uma tarefa, provendo assim uma maior flexibilidade para o usuário. Ao mesmo tempo, a solução permite que essa flexibilidade se realize dentro da própria interface de uso do sistema, evitando portanto a passagem do usuário de um ambiente de uso para um ambiente de extensão distinto.

Nossa proposta se diferencia dessa no sentido em que ela apresenta ao usuário as estratégias que o designer da aplicação julgou adequadas para atingir-se um objetivo. Caso essas estratégias não satisfaçam o usuário, ele pode construir suas próprias estratégias no ambiente de extensão. Essa distinção favorece abordagens diferentes de design de interação. Enquanto nossa proposta aborda uma resolução de problema e design de interface pré-definidos, que guiam passo-a-passo os usuários em direção à solução (como o design de *wizards*), a proposta de [Dourish et al.'1996] implica em uma abordagem de resolução de problema do tipo “faça você mesmo” e um design de interface orientado a objetos, no qual o usuário decide qual será o próximo passo realizado.

Por outro lado o trabalho de [Dourish et al.'1996] não discute os processos de comunicação relacionados à solução proposta (e.g. como usuários compartilhariam suas estratégias). Note-se que usuários poderiam adotar sempre formas diferentes para solucionar um mesmo problema. Em decorrência disso, usuários teriam discursos pessoais diferentes sobre o uso da aplicação e não necessariamente haveria um discurso compartilhado entre eles. Logo, durante um processo de comunicação, os usuários precisariam construir uma base de conhecimento compartilhada através do reconhecimento de interseções entre seus discursos pessoais.

Conseqüentemente haveria um custo maior nas comunicações do que se houvesse ao menos um discurso compartilhado.

Cabe lembrar também que é possível que, apesar da flexibilidade oferecida pela proposta [Dourish et al.'1996], ainda haja outros casos em que se queira modificar o sistema original (e.g. a alteração de restrições que a princípio não poderiam ser quebradas). A proposta também não discute essa questão e naturalmente não discute os processos comunicativos relacionados à mesma.

[Jørgensen'2001] propõe um *framework* para modelagem de workflows flexíveis que foge à noção convencional de modelos de processo formalmente completos e consistentes. Partindo da visão de modelos de processo como mediadores de conhecimento, ele dirige sua atenção para como os processos estão articulados e para como o conhecimento representado no modelo é ativado. Ele discute que a maioria das pesquisas de workflows adaptáveis se baseia na premissa de que a interpretação do modelo de workflow é feita **somente** pelo seu motor de ativação. Seguindo uma semântica de ativação automática, o sistema atua como um impositor do 'script' prescrito pelo modelo. Ou seja, usuários contribuem fazendo alterações ao modelo, mas não interpretando qualquer parte dele (durante sua ativação). Logo, o modelo deve estar formalmente completo para prevenir ambigüidades e/ou paralisações no processo.

[Jørgensen'2001] propõe que outras semânticas de ativação de modelos de workflow sejam investigadas. Ele argumenta em favor de uma semântica de ativação interativa em que o computador e os usuários cooperam na interpretação do modelo durante sua ativação, de forma situada. O computador toma decisões de acordo com os fragmentos prescritos enquanto que os usuários decidem sobre ambigüidades. O autor compara a diferença entre ativação automática e interativa à diferença entre máquinas de Turing e máquinas de interação [Wegner'1997, Wegner&Goldin'1999]. A principal característica das máquinas de interação é o poder de questionar atores humanos (usuários) durante a sua computação. Neste caso, mudanças ao modelo são acomodadas não só através da atualização do mesmo, mas também através da ativação (ou re-interpretação) controlada pelo usuário. Assim, a ativação de uma tarefa pode não estar prescrita na representação do workflow, sendo então disparada pelo próprio usuário.

Para isso, o usuário tem acesso a uma representação que mostra a rede de tarefas do workflow. O usuário pode escolher qual das tarefas deseja realizar. O sistema permite que o usuário indique se irá ou não realizar os pré-requisitos da tarefa escolhida. Depois de realizada a tarefa, sistema segue a seqüência de tarefas prescrita na representação vigente, mas o usuário sempre pode decidir se quer ou não seguir essa seqüência.

A proposta de [Jørgensen'2001], embora não parta de estudos de EUP nem da teoria Semiótica (e portanto não trate de conceitos como os de semiose ilimitada ou o de semiótica específica), pode ser considerada similar à nossa no sentido em que parte da premissa de que todo modelo é incompleto e pode mudar durante sua ativação. Em ambas as propostas prevê-se a possibilidade de modificações ao modelo, assumindo-se que qualquer coisa (não fixada explicitamente) pode mudar. Através de processos de interação, os usuários podem modificar o modelo de forma não pré-determinada por desenvolvedores, acomodando assim interpretações situadas do modelo, não-determinismos e incertezas inerentes às realidades organizacionais.

Vimos que a proposta de [Jørgensen'2001] e a nossa compartilham premissas básicas, apesar de seus *backgrounds* distintos e complementares entre si. Ainda assim as soluções propostas pelo nosso trabalho e o de [Jørgensen'2001] são diferentes. A relação de entre essas duas soluções é similar à relação do nosso trabalho com o de [Dourish et al.'1996]. Por exemplo, na solução proposta por [Jørgensen'2001] não há, a princípio, uma distinção entre o ambiente de uso e o ambiente de extensão, como é o caso em nossa proposta. Na proposta de [Jørgensen'2001] um usuário modifica o modelo do workflow em tempo de execução, atuando sobre uma representação do mesmo.

Testes iniciais da solução de [Jørgensen'2001] indicaram que a representação do modelo apresentada na interface ainda parece muito complicada para alguns usuários. Já o nosso trabalho lida com este problema pois apóia-se no conhecimento que os usuários têm da UIL para a construção de extensões ao workflow, além de possivelmente prover também mais de uma representação de um mesmo workflow.

A proposta de [Jørgensen'2001] permite que usuários modifiquem o modelo do workflow, combinando as tarefas existentes de diferentes formas através de uma interface. Ainda assim, não fica claro se é permitido aos usuários criar tarefas novas que, por exemplo, combinem estruturas de informação de forma diferente da prevista, como é permitido em nossa proposta. Assim como no trabalho de [Dourish et al.'1996], a solução de [Jørgensen'2001] implica em uma abordagem de resolução de problema do tipo “faça você mesmo”, no qual o usuário decide qual será o próximo passo realizado. Já o nosso trabalho favorece uma resolução de problema e design de interface pré-definidos (como o design de *wizards*).

Em trabalhos futuros pretendemos, através de estudos das pesquisas sobre cognição distribuída [Hollan et al.'2000, Wright et al.'2000], enriquecer as discussões aqui apresentadas, especialmente aquela sobre representações a serem apresentadas aos usuários finais e suas conseqüências sobre estratégias de resolução de problemas. A cognição distribuída estuda interações entre pessoas e recursos ou materiais no ambiente. Ela observa como representações do mundo material fornecem oportunidades para reorganizar o sistema cognitivo distribuído. A distribuição do conhecimento se dá tanto entre um usuário e uma representação computacional de sua tarefa, no âmbito estrito da Interação Humano-Computador, como entre diferentes usuários e uma representação computacional, no âmbito do trabalho cooperativo apoiado por computador (ou CSCW - *Computer-Supported Collaborative Work*). Por exemplo o modelo de recursos proposto por [Wright et al.'2000] apresenta dimensões que permitem avaliar soluções para a efetivação e construção de planos e que portanto podem servir de base para uma comparação entre diferentes propostas de workflows flexíveis.

Nesse capítulo mostramos como nosso trabalho se relaciona com seus semelhantes e representa avanços em diferentes aspectos das pesquisas de EUP e de sistemas de workflow flexíveis realizadas até o presente. O trabalho preenche lacunas ainda não exploradas por essas pesquisas de forma motivada pelos resultados apresentados pelas mesmas.

## 7. Conclusão e Trabalhos Futuros

O principal diferencial desse trabalho reside em uma abordagem direta e teoricamente fundamentada dos processos comunicativos sobre extensões. Suas principais contribuições são um modelo semiótico desses processos comunicativos e uma linguagem para representação de discursos sobre extensões baseada no modelo.

O modelo caracteriza, organiza e explica fenômenos de comunicação sobre extensões, permitindo assim uma melhor compreensão das questões relacionadas aos mesmos. Como resultado dessa compreensão, aponta-se para diretrizes de design que podem ser adotadas em aplicações de grupo extensíveis de forma a apoiar a realização de comunicações eficazes e eficientes. Mostra-se por exemplo a necessidade de comunicações bidirecionais e preferencialmente assíncronas e argumenta-se sobre a utilização da aplicação original como base de conhecimento compartilhada entre os agentes comunicantes e, portanto, sobre a utilização de representações para expressão de extensões baseadas na linguagem de interface.

Vimos que com base nesse modelo pode-se prover referências descritivas para a especificação, implementação, discussão e revisão de projetos de sistemas extensíveis que abordam de forma integrada o aspecto da criação de uma cultura de adaptação, visando não só a disponibilização de um gradiente de diferentes mecanismos de EUP, mas também o tratamento explícito dos processos de comunicação que apóiam a formação de uma cultura.

Logo, podemos dizer que o modelo oferece uma base para futuros métodos e ferramentas que pretendam atuar como facilitadores dos processos e atividades em que os usuários se envolvem ao estender uma aplicação.

Com base no modelo, o trabalho oferece uma linguagem formalizada para representação de discursos que serve como referencial comum para outras representações a serem utilizadas por usuários nos processos de comunicação sobre extensões. A linguagem integra diferentes aspectos de uma aplicação adotando uma perspectiva de interação, permitindo assim que usuários utilizem seus conhecimentos sobre a UIL para expressar extensões. Mais ainda, a

partir da aplicação do princípio do contínuo semiótico, a linguagem apresenta estruturas sintáticas que garantem o ciclo mínimo de interação das extensões. Ou seja, garante-se a expressão abstrata de extensões no código da UIL que, por ser o código artificial preponderante entre os agentes comunicantes, tende a maximizar a eficácia e eficiência nas comunicações sobre extensões.

Cabe notar que através dessa linguagem nossa proposta visa a construção de mensagens formais e interativas por usuários finais, constituindo assim um incentivo ao aumento do uso do computador como mídia [Kammersgaard'1988, Andersen et al.'1993] que, nesse sentido, ainda permanece sub-explorado pelos usuários, dado os conhecimentos de programação que a construção dessas mensagens requer [diSessa&Abelson'1986].

A avaliação empírica da nossa proposta e o estudo de pesquisas correlatas mostrou que ainda há uma série de trabalhos futuros a serem realizados. Um trabalho importante é o design, implementação e avaliação de um sistema extensível funcional que adote nosso modelo e linguagem de representação. Assim poderemos avaliar mais profundamente esse trabalho e seus impactos sobre o projeto de sistemas extensíveis e sobre o desenvolvimento de uma cultura de adaptação. Será interessante observar também o impacto de nossa proposta sobre práticas de manutenção e até mesmo de avaliação de um software, já que ela motiva uma aproximação entre designers e usuários.

Pretendemos investigar a existência de *design patterns* ou padrões de design de extensões que possam informar o desenvolvimento de futuros sistemas extensíveis. Em termos semióticos, podemos dizer que estaremos observando fenômenos de hipocodificação e hipercodificação. Ou seja, estaremos observando tendências ou preferências na formação de representações de extensões, assim como restrições sociais ou culturais sobre o conjunto total de representações possíveis de se produzir dentro do que é permitido [de Souza'a ser publicado]. Conseqüentemente estaremos buscando padrões de evolução de aplicações extensíveis por usuários finais.

Um aspecto que ainda não tratamos em nosso modelo e que está sendo abordado por pesquisas em andamento da Engenharia Semiótica é o fato de que uma aplicação é uma



mensagem que reflete, ou pelo menos deveria refletir, o consenso de um grupo de designers. O tratamento dessa questão foi mantido fora do escopo desse trabalho dada a sua complexidade, mas deverá ser incorporado em nosso modelo para que se espelhe uma mensagem que é derivada dessa realidade e depois transmitida a partir de um software.

Poder-se-ia explorar mais profundamente o elemento de contexto do modelo, com destaque para o *background* dos agentes comunicantes. Seria interessante por exemplo estudar aspectos de representação e apresentação de informações sobre o perfil dos agentes comunicantes durante os processos de comunicação de forma a enriquecer esses processos e torná-los mais eficazes e eficientes.

Considerando-se que estamos visando aplicações extensíveis de sistemas orientados a tarefas para a WWW, um trabalho a ser realizado seria o estudo de como a linguagem de representação poderia acomodar modelos ricos de navegação, como os que podem ser projetados a partir da OOHDM [Rossi et al.'1995, Schwabe et al.'1996, Schwabe&Rossi'1998]. Além disso, seria interessante averiguar também se e como a linguagem acomodaria uma variação nos modelos que são por ela integrados (e.g. outros modelos de informação, interface, etc.).

Antecipamos também a necessidade de um estudo e derivação de propostas de tratamento sistemático da tensão existente entre a necessidade de se construir extensões e, ao mesmo tempo, manter a aplicação original funcional como base de conhecimento compartilhada entre os agentes comunicantes. Em nossa investigação empírica, observamos exemplos de extensão que poderiam afetar a mensagem original do designer e que só poderiam ser detectados através de uma análise semântica e baseada em conhecimento sobre o domínio da aplicação. Era o caso por exemplo da re-utilização da data de devolução para fins de histórico, que mudaria a semântica atribuída à essa informação e, como uma de suas conseqüências, tornaria a visualização de vencimentos inconsistente.

Como no nível lingüístico da X-DIS não há diferenciação entre essa extensão e outras perfeitamente úteis e consistentes com a aplicação original (e.g. a diminuição de prazo de devolução de um empréstimo, que também atualiza a informação de data de devolução), é

preciso criar regras semânticas e mecanismos inteligentes que atuem no ambiente de extensão de forma a garantir que serão expressos na X-DIS somente textos que não ferem a mensagem original do designer.

Além de proteger a mensagem do designer, é preciso proteger também a mensagem do estendedor passada através de uma extensão, no sentido de evitar a construção de extensões conflitantes, como apontado por nossa investigação empírica e também no trabalho de [Malone et al.' 1995]. Esse é o caso de extensões independentes que, quando assim utilizadas, não causam problemas, mas que quando combinadas introduzem inconsistência à aplicação. Por exemplo pode-se utilizar uma extensão de prorrogação de prazo de empréstimo e, depois, querer-se utilizar uma extensão para reserva de material. Nesse caso, seria necessário observar se a extensão de prorrogação de prazo precisa ser modificada, de forma a evitar a prorrogação caso haja uma reserva. Mais uma vez, para evitar esse tipo de problema é necessário um tratamento semântico, baseado em conhecimento do domínio, durante o processo de construção e adoção de extensões.

Vislumbramos também uma extensão do nosso modelo a partir de pesquisas sobre formas de apoio à expressão formalizada de intenções de comunicação e de posições na negociação de ações. Prevemos que a incorporação de soluções a esse respeito podem servir como ponto de partida para o tratamento da organização e recuperação de um histórico de comunicações sobre extensões. Antecipamos também que o armazenamento de histórico poderá servir, a longo prazo, como uma plataforma para o estabelecimento de relações entre o comportamento evolutivo de sistemas de software e os padrões de evolução de sistemas semióticos. Conseqüentemente, seria reforçada a argumentação de que o processo de evolução de software está intrinsecamente relacionado a aspectos semióticos. Poderia haver também um enriquecimento nos estudos sobre o comportamento de sistemas semióticos, tendo que seria possível observar padrões de evolução dos mesmos através das extensões dos significados de um domínio de aplicação.

## 8. Referências

[Abbott&Sarin'1994] Abbott, K. R. e Sarin, S. K. (1994) "Experiences with Workflow Management: Issues for the Next Generation". Computer Supported Cooperative Work'94 (CSCW'94). Págs. 113-120. ACM Press.

[Andersen et al.'1993] Andersen, P. B., Holmqvist, B. e Jensen, J. (eds.) (1993) "Computers as Media". Cambridge University Press, UK.

[Andersen'2001] Andersen, P. B. (2001) "What Semiotics Can and Cannot Do for HCI". Knowledge Based Systems - Special Issue. Vol. 14. Issue 8. Elsevier Science. Págs. 419-424.

[Barbosa'1999] Barbosa, S. D. J. (1999) "Programação Via Interface". Tese de Doutorado. Orientadora: Clarisse Sieckenius de Souza. Departamento de Informática. PUC-Rio. Rio de Janeiro, RJ.

[Barbosa&de Souza'1999] Barbosa, S. D. J. e de Souza, C. S. (1999) "Making More Sense out of Users' Utterances". II Workshop sobre Fatores Humanos em Sistemas Computacionais - IHC'99 Proceedings.

[Barbosa&de Souza'2000] Barbosa, S. D. J. e de Souza, C. S. (2000) "Extending Software Through Metaphors and Metonymies". IUI'2000 (Intelligent User Interfaces'2000).

[Barbosa et al.'2001] Barbosa, C. M. A., Prates, R. O. e de Souza, C. S. (2001) "Analisando a Comunicação entre Usuários em Ambientes de Grupo". IV Workshop sobre Fatores Humanos em Sistemas Computacionais - IHC'2001 Proceedings. SBC. Págs. 25-35.

[Booch et al.'1999] Booch, G., Rumbaugh, J. e Jacobson, I. (1999) "The Unified Modeling Language User Guide". Addison-Wesley.

[Brown&Yule'1983] Brown, G. e Yule, G. (1983) "Discourse Analysis". Cambridge University Press.

[Chang'1990] Chang, S. (1990) "Principles of Visual Programming Systems" Prentice-Hall. Englewood Cliffs, New Jersey.

[Clark&Brennan'1991] Clark, H. H. e Brennan, S. E. (1991) "Grouding in Communication". Em Resnick, L. B., Levine, J. M. e Teasley, S. D. (eds.) *Socially Shared Cognition*. American Psychological Association. Washington, DC.

[Clement'1990] Clement, A. (1990) "Cooperative Support for Computer Work: A Social Perspective on the Empowering of End Users". Proceedings of the Conference on Computer Supported Cooperative Work'90 (CSCW'90). ACM Press. Págs. 223-236.

[Conklin&Begeman'1988] Conklin, J. e Begeman, M. (1988) "gIBIS: A Tool for Exploratory Policy Discussion". ACM Transactions on Information Systems. Vol. 6. No. 4. Págs. 303-331. ACM.

[Cunha et al.'2000a] Cunha, C. K. V., de Souza, C. S., Quental, V. S. T. D. B. e Schwabe, D. (2000) "A Model for Extensible Web-based Information Intensive Task Oriented Systems". Relatório Técnico. Monografias em Ciência da Computação. C.J.P. de Lucena (ed.). PUC-RioInf MCC04/00. Departamento de Informática., PUC-Rio.

[Cunha et al.'2000b] Cunha, C. K. V., de Souza, C. S., Quental, V. S. T. D. B. e Schwabe, D. (2000) "A Model for Extensible Web-based Information Intensive Task Oriented Systems". Proceedings do HCI2000 (Human Computer Interaction 2000). Springer-Verlag. Págs. 205-219.

[Cypher'1993] Cypher, A. (eds.) (1993) "Watch What I Do: Programming by Demonstration". The MIT Press.

[Dahis'2001] Dahis, G. (2001) "Um Modelo para a Especificação de Cenários de Interação". Dissertação de Mestrado. Orientadora: Clarisse Sieckenius de Souza. Departamento de Informática. PUC-Rio. Rio de Janeiro, RJ.

[de Souza'1993] de Souza, C. S. (1993) "The Semiotic Engineering of User Interface Languages". *International Journal of Man-Machine Studies*. No. 39. Págs. 753-773.

[de Souza'1997] de Souza, C. S. (1997) "Supporting End-User Programming with Explanatory Discourse". *Proceedings of ISAS'97 - Intelligent Systems and Semiotics'97 - A Learning Perspective*. NIST. Págs. 461-466.

[de Souza et al.'2001] de Souza, C. S., Barbosa, S. D. J. e Silva, S. R. P. (2001) "Semiotic Engineering Principles for Evaluating End-User Programming Environments". *Interacting with Computers*. Vol 13. Elsevier Science. Págs. 467-495.

[de Souza'a ser publicado] de Souza, C. S. (a ser publicado) "Semiótica e Engenharia Semiótica". Relatório Técnico.

[DiGiano&Eisenberg'1995] DiGiano, C. and Eisenberg, M. (1995) "Self-disclosing Design Tools: A Gentle Introduction to End-User Programming". *Proceedings do DIS'95 (Symposium on Designing Interaction Systems)*. ACM Press. Págs. 189-197.

[diSessa&Abelson'1986] diSessa, A. A. e Abelson, H. (1986) "Boxer: A Reconstructible Computational Medium". *Communications of the ACM*. Vol. 29. No. 9. Págs. 859-868.

[Dourish et al.'1996] Dourish, P., Holmes, J., MacLean, A., Marquardsen, P. e Zbyslaw, A. (1996) "Freeflow: Mediating between Representation and Action in Workflow Systems". *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work (CSCW'96)*. Págs. 190-198. ACM Press.

[Dourish'1997] Dourish, P. (1997) "Accounting for System Behavior: Representation, Reflection, and Resourceful Action". Em Kyng, M. e Mathiassen, L. (eds.) *Computers and Design in Context*. The MIT Press.

[Draper'1986] Draper, S. A., (1986) "Display Managers as the Basis for User-Machine Communication". Em Norman e Draper (eds.) *User-Centered System Design*. Lawrence Erlbaum and Associates.

[Eco'1976] Eco, U. (1976) "A Theory of Semiotics". Indiana University Press.

[Eco'1984] Eco, U. (1984) "Semiotics and the Philosophy of Language". Indiana University Press.

[Eisenberg'1994] Eisenberg, M. e Fischer, G. (1994) "Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance". Proceedings of CHI'94 (Human Factors in Computing Systems). ACM Press. Págs. 431-437.

[Eisenberg'1995] Eisenberg, M. (1995). "Programmable Applications". SIGCHI Bulletin. Vol. 27. No. 2. ACM.

[Ellis et al.'1991] Ellis, C., Gibbs, S. e Rein, G. (1991) "Groupware: Some Issues and Experiences". Communications of the ACM. Vol. 34. No. 1. Págs. 39-58.

[Gantt&Nardi'1992] Gantt, M. e Nardi, B. (1992) "Gardeners and Gurus: Patterns of Cooperation among CAD Users". Proceedings of CHI'92. ACM Press. Págs. 107-117.

[Glance et. al.'1996] Glance, N. S., Pagani, D. S. e Pareschi, R. (1996) "Generalized Process Structure Grammars (GPSG) for Flexible Representations of Work". Computer Supported Cooperative Work'96 (CSCW'96). Págs. 180-189. ACM Press.

[Greif'1992] Greif, I. (1992) "Designing Group-Enabled Applications: A Spreadsheet Example". Em Coleman, D. (ed.). *Groupware'92*. Morgan Kaufmann. Págs. 515-525.

[Grice'1975] Grice, H. P. (1975) "Logic and Conversation". Em Cole, P. e Morgan, J. L. (eds.) *Syntax and Semantics*. Vol. 3 (Speech Acts), 225-242. Academic Press.

[Harel'1988] Harel, D. (1988) "On Visual Formalisms". *Communications of the ACM*. Vol. 31. No. 5. Págs. 514-530.

[Hollan et al.'2000] Hollan, J., Hutchins, E. e Kirsh, D. (2000) "Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research". *ACM Transactions on Computer-Human Interaction* Vol. 7. No. 2. Págs. 174-196.

[ISO\_WG17'1995] ISO/IEC JTC1/SC22/WG17 International Standardization Working Group for the Programming Language Prolog. "ISO/IEC 13211-1 Programming Language Prolog Part 1". URL:

<http://www.sju.edu/~jhodgson/wg17/wg17web.html> (última visita em 28/01/2002)

[Jakobson'1960] Jakobson, R. (1960) "Closing Statements: Linguistics and Poetics". Em Sebeok, T. (ed.) *Linguistics and Communication*. MIT Press. New York, NY.

[Johansen'1991] Johansen, R. (1991) "Leading Business Teams". Addison-Wesley.

[Johnson et al.'1993] Johnson, J., Nardi, B., Zamer, C. e Miller, J. (1993) "ACE: A New Approach to Building Interactive Graphical Applications". *Communications of the ACM*. Vol. 36. April.

[Jørgensen'2001] Jørgensen, H. D. (2001) "Interaction as a Framework for Flexible Workflow Modelling". *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*. ACM Press. Págs. 32-41.

[Kammersgaard '1988] Kammersgaard, J. (1988) "Four Different Perspectives on Human-Computer Interaction". *International Journal of Man-Machine Studies*. No. 28. Págs. 343-362.

[Laufer&Fuks'1995] Laufer, C. C. e Fuks, H. (1995) "ACCORD: Conversation Clichés for Cooperation". *Proceedings of The International Workshop on the Design of Cooperative Systems (COOP'95)*. Págs. 351-369.

[Leite'1998] Leite, J. (1998) "Modelos e Formalismos para a Engenharia Semiótica de Interfaces de Usuário". Tese de Doutorado. Orientadora: Clarisse Sieckenius de Souza. Departamento de Informática. PUC-Rio. Rio de Janeiro, RJ.

[Levine&Moreland'1991] Levine, J. M. e Moreland, R. L. (1991) "Culture and Socialization in Work Groups". Em Resnick, L. B., Levine, J. M. e Teasley, S. D. (eds.) *Socially Shared Cognition*. American Psychological Association. Washington, DC.

[Lieberman'2001] Lieberman, H. (eds.) (2001) "Your Wish is My Command - Programming by Example". Morgan Kaufmann.

[Mackay'1990] Mackay, W. (1990) "Patterns of Sharing Customizable Software". *Proceedings of the Conference on Computer Supported Cooperative Work'90 (CSCW'90)*. ACM Press. Págs. 209-221.

[MacLean et al.'1990] MacLean, A., Carter, K., Lövsstrand, L. e Moran, T. (1990) "User-Tailorable Systems: Pressing the Issues with Buttons". *Proceedings of CHI'90 (Human Factors in Computing Systems)*. ACM Press. Págs. 175-182.

[McDaniel'2001] McDaniel, R. (2001) "Demonstrating the Hidden Features that Make an Application Work". Em Lieberman, H. (eds.) *Your Wish is My Command - Programming by Example*. Capítulo 8. Morgan Kaufmann.



[Malone et al.'1995] Malone, T. W., Lai, K. e Fry, C. (1995) "Experiments with Oval: A Radically Tailorable Tool for Cooperative Work". *ACM Transactions on Information Systems*. Vol. 13. No. 2. Págs. 177-205.

[Myers'1992] Myers, B. A. (1992) "Languages for Developing User Interfaces". Jones and Bartlett Publishers, Inc.

[Myers'2000] Myers, B. A. (2000) "Intelligence in Demonstrational Interfaces". *Communications of the ACM*. Vol. 43. No. 3. Págs. 82-89.

[Nadin'1988] Nadin, M. (1988) "Interface Design: A Semiotic Paradigm". *Semiotica*. Vol. 69 (3/4).

[Nardi&Miller'1990] Nardi, B. e Miller, J. (1990) "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development". *International Journal of Man-Machine Studies*. Vol. 34. Págs. 161-184.

[Nardi'1993] Nardi, B. (1993) "A Small Matter of Programming". The MIT Press.

[O'Malley'1986] O'Malley, C. E. (1986) "Helping Users Help Themselves". Em Norman, D. A. e Draper, S. (eds.) *User Centered System Design*. Capítulo 18. Lawrence Erlbaum and Associates.

[Ousterhout'1998] Ousterhout, J. K. (1998) "Scripting: Higher-Level Programming for the 21<sup>st</sup> Century". *Computer*. Vol. 31. No. 3. págs. 23-30. IEEE.

[Peirce'1931] Peirce, C. S. (1931) "Collected Papers". Harvard University Press.

[Pereira&Warren'1980] Pereira, F. C. N. e Warren, D. H. D. (1980) "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with

Augmented Transition Networks”. *Artificial Intelligence*. Vol 13. págs. 231-278. North-Holland Publishing Company.

[Prates'1998] Prates, R. O. (1998) “A Engenharia Semiótica de Linguagens de Interfaces Multi-Usuário”. Tese de Doutorado. Orientadora: Clarisse Sieckenius de Souza. Departamento de Informática. PUC-Rio. Rio de Janeiro, RJ.

[Prates&de Souza'1998] Prates, R. O. and de Souza, C.S. (1998) “Towards a Semiotic Environment for Supporting the Development of Multi-User Interfaces”. Fourth International Workshop on Groupware (CRIWG'98). págs. 53-67.

[Preece et al.'1994] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. e Carey, T. (1994) *Human-Computer Interaction*. Addison-Wesley, Wokingham, UK.

[Rader et al.'1998] Rader, C., Cherry, G., Repenning, A. e Lewis, C. (1998) “Principles to Scaffold Mixed Textual and Iconic End-User Programming Languages”. Proceedings of 1998 IEEE Symposium of Visual Languages. págs. 187-194.

[Repenning et al.'1999] Repenning, A., Ioannidou, A. e Philips, J. (1999) “Collaborative Use & Design of Interactive Simulations”. Proceedings of Computer-Supported Collaborative Learning (CSCL'99).

[Rossi&Schwabe'1998] Rossi, G. e Schwabe, D. (1998) “An Object-Oriented Approach to Web-Based Application Design”. *Theory and Practice of Object Systems*. Vol. 4. No. 4. págs.207-225. Wiley and Sons.

[Rossi et al.'1995] Rossi, G., Schwabe, D., Lucena, C.J.P. e Cowan, D.D. (1995) “An Object-Oriented Model for Designing the Human-Computer Interface of Hypermedia Applications”. Proceedings of the International Workshop on Hypermedia Design'95 (IWHD'95). Springer Verlag.

[Schwabe et al.'1996] Schwabe, D., Rossi, G. e Barbosa, S.D.J. (1996) "Systematic Hypermedia Application Design with OOHDM". Seventh ACM Conference on Hypertext (Hypertext'96).

[Searle'1979] Searle, J. R. (1979) "Expression and Meaning". Cambridge University Press.

[Shneiderman'1998] Shneiderman, B. (1998) "Designing the User Interface". Addison-Wesley. Terceira edição.

[Shneiderman'2001] Shneiderman, B. (2001) Foreword. Em Lieberman, H. (eds.) *Your Wish is My Command - Programming by Example*. Morgan Kaufmann.

[Silva'2001] Silva, S. R. P. (2001) "Um Modelo Semiótico para Programação por Usuários Finais". Tese de Doutorado. Orientadora: Clarisse Sieckenius de Souza. Departamento de Informática. PUC-Rio. Rio de Janeiro, RJ.

[Silveira et al.'2000] Silveira, M. S., Barbosa, S. D. J. e de Souza, C. S. (2000) "Modelo e Arquitetura de Sistemas de *Help Online*". III Workshop sobre Fatores Humanos em Sistemas Computacionais - IHC'2000 Proceedings.

[Suchman'1987] Suchman, L. (1987) "Plans and Situated Actions". Cambridge University Press.

[Wellner'1989] Wellner, P. D. (1989) "A UIMS Based on Statecharts for Prototyping and Target Implementation". Proceedings of CHI'89 (Human Factors in Computing Systems). ACM Press. Págs. 177-182.

[Wegner'1997] Wegner, P. (1997) "Why Interaction Is More Powerful than Algorithms". Communications of the ACM. Vol. 40. No. 5. Págs. 80-91. ACM.

[Wegner&Goldin'1999] Wegner, P. e Goldin, D. (1999) "Interaction as a Framework for Modeling". Conceptual Modeling. Current Issues and Future Directions. LNCS 1565. Springer.

[WfMC'1995] Workflow Management Coalition – The Workflow Reference Model; Document Number WFMC-TC00-1003, Document Status – Issue 1.1; Hollingsworth, D.; 1995.

[WfMC'1999] Workflow Management Coalition – Terminology and Glossary; Document Number WFMC-TC-1011, Document Status – Issue 3.0; 1999.

[Wielemaker'2001] Wielemaker, J. (2001) "SWI-Prolog Home". URL: <http://www.swi.psy.uva.nl/projects/SWI-Prolog/> (última visita em 6 de Junho de 2001)

[Winograd'1987] Winograd, T. (1987) "A Language/Action Perspective on the Design of Cooperative Work". Human-Computer Interaction. Vol 3. Págs. 3-30.

[Woods'1970] Woods, W. A. (1970) "Transition Network Grammars for Natural Language Analysis". Communications of the ACM. Vol. 13. No. 10. Págs. 591-606. ACM.

[Wright et al.'2000] Wright, P., Fields, R. E. e Harrison, M. D. (2000) "Analyzing Human-Computer Interaction as Distributed Cognition: The Resources Model". Human-Computer Interaction. Vol. 15. Págs. 1-41. Lawrence Erlbaum Associates.

[Zarmer&Chew'1992] Zarmer, C. e Chew, C. (1992) "Frameworks for Interactive, Extensible, Information-Intensive Applications". Proceedings of UIST'92.

[Zarmer et al.'1992] Zarmer, C., Nardi, B., Johnson, J. e Miller, J. (1992) "ACE: Zen and the Art of Application Building". Proceedings of the 25<sup>th</sup> Hawaii International Conference on System Sciences. Vol. 2. págs. 687-698.

## Apêndice I. Implementação da X-DIS

Para a implementação da linguagem formal foi utilizada a representação em DCG (*Definite Clause Grammars*) [Pereira&Warren'1980] que permite a expressão de gramáticas em lógica de primeira ordem que constituem programas práticos efetivos, claros e modulares da linguagem Prolog. O interpretador Prolog utilizado foi o SWI-Prolog [Wielemaker'2001] na sua versão 3.4.5 que segue o padrão ISO Prolog (Parte 1) [ISO\_WG17'1995].

```

register(tmpvariable_or_navigationalitem_or_nltextitem_or_infoitem).

/* estrutura proveniente do modelo de interface com todas as informações
sobre ela */
load_interface_structure(buildq(interface_structure,list_of_registers_list_
of_forms)).

/* leitura de input do usuário */
read_user_input(buildq(interface_structure,read_input)).

/* leitura de trigger do usuário */
read_selected_trigger(read_selected_trigger_by_user).

/* tela de entrada de dados */
generate_input_screen(setr_register_interfaceName_generateHtmlForm_params_i
nterfaceInputFieldsStruct).

/* tela de saída de dados que é um estado final ou admite fechamento /
navegação para outra tela */
generate_output_screen(setr_register_generate_html_text_ou_visualizacao_par
ams).

form(getr(X)):- register(X).
form(getdb(dbItem)).
form(valor_fixo). /* atribui valor constante */
form(buildq(tmpvar_or_navigationalitem_or_txtgen,list_of_registers_and_list
_of_forms)).
form(list_of_forms).
/* ações sobre objetos, geração de texto e de âncora */
form(quote(executable_code)).

category(output_message_or_object_visualization_or_menu).

/* ***** As ações (actions) iguais diminuem o número de arcos gerados
para análise. Para a geração completa, deve-se utilizar actions diferentes
***** */

```

```

arc_io(cat(Category,list_of_tests,action(A),generate_output_screen(B),
read_trigger(C), action(A), terminate_int)) :- category(Category),
action(A), generate_output_screen(B), read_trigger(C), action(A).

arc_io(cat(data_entry,list_of_tests,action(A),load_input_fields(D),generate
_input_screen(B), read_input_take_action(C), read_trigger(E),
terminate_int)) :- action(A),load_input_fields(D),
generate_input_screen(B), read_input_take_action(C), read_trigger(E).

arc_inicial(push(to_state,list_of_tests,action(A), terminate_inicial)) :-
action(A).

arc_inicial(tst(label,list_of_tests,action(A),generate_output_screen(B),
read_trigger(C), action(A), terminate_inicial )):- action(A),
generate_output_screen(B), read_trigger(C), terminate(E).

arc_inicial(tst(label,list_of_tests,action(A),terminate_inicial )):-
action(A).
/* pode voltar p/ a tela anterior c/ ou s/ mensagem de erro, ao invés de
gerar outra tela */

arc_inicial(pop(X,list_of_tests)):- form(X).

arc_inicial(parallel(list_of_terminates_iniciais)).

arc_inicial(loop_state_inicial).

arc_inicial(nenhum_comeca_direto_com_input_output).

arc_out(cat(Category,list_of_tests,action(A),generate_output_screen(B),
read_trigger(C), action(A), terminate_final_ou_retorno)) :-
category(Category), action(A), generate_output_screen(B), read_trigger(C),
action(A).

arc_int(cat(data_entry,list_of_tests,action(A),load_input_fields(D),generat
e_input_screen(B), read_input_take_action(C), read_trigger(E),
terminate_int)) :- action(A),load_input_fields(D),
generate_input_screen(B), read_input_take_action(C), read_trigger(E).

arc_int(cat(Category,list_of_tests,action(A),generate_output_screen(B),
read_trigger(C), action(A), terminate_int)) :- category(Category),
action(A), generate_output_screen(B), read_trigger(C), action(A).

arc_int(push(to_state,list_of_tests,action(A), terminate_int)) :-
action(A).

arc_int(tst(label,list_of_tests,action(A),generate_output_screen(B),
read_trigger(C), action(A), terminate_int )):- action(A),
generate_output_screen(B), read_trigger(C).

arc_int(tst(label,list_of_tests,action(A),terminate_int )):- action(A).
/* pode voltar p/ a tela anterior c/ ou s/ mensagem de erro, ao invés de
gerar outra tela */

arc_int(pop(X,list_of_tests)):- form(X).

arc_int(parallel(list_of_terminates)).

arc_int(loop_state_int).

```

```

arc_fim_retorno(fim).
arc_fim_retorno(retorna_estado_int).

/* validação de dados, regras de negócio, etc. */
post_test(arbitrary_post_condition).
test(arbitrary_pre_condition).
test(input_validation).

/* checa identificação do usuário e controle de acesso a objetos, funções,
etc. */
test(security_id_validation_or_control_access).

test(synchronize).
test(communicate).
test(monitor).

action(setr(Register,Form)) :- register(Register), form(Form).
action(sendr(Register,Form)) :- register(Register), form(Form).
action(liftr(Register,Form)) :- register(Register), form(Form).
action(setdb(dbOper, dbItems, Form)) :- form(Form).
action(zero_or_more_actions).

load_input_fields(setr(interface_input_fields, Form)) :-
load_interface_structure(Form).

/* lê input de dado da interface */
read_input_take_action(setr(interface_input_fields,Form)) :-
read_user_input(Form).

/* (read_input_fields action)* */
read_input_take_action(sendr(interface_input_fields,Form)) :-
read_user_input(Form).

read_input_take_action(liftr(interface_input_fields,Form)) :-
read_user_input(Form).

read_input_take_action(one_or_more_read_input_and_actions).

/* lê input de trigger da interface: botão, link ou fechamento da janela */
/* link ativado (aponta para representação no modelo do domínio */
read_trigger(setr(selected_trigger,read_user_trigger)).
read_trigger(sendr(selected_trigger,read_user_trigger)).
read_trigger(liftr(selected_trigger,read_user_trigger)).

arcset(arcst(state_inicial,Ini, le_in_out, Io, state_int, Int, le_out, Out,
fim_ou_tem_mais, FimOuTemMais)) :- arc_inicial(Ini), arc_io(Io),
arc_int(Int), arc_out(Out), arc_fim_retorno(FimOuTemMais), nl,
write('state_inicial_top: '), write(Ini), nl, write(Io), nl,
write('state_int: '), write(Int), nl, write(Out), nl,
write('fim_ou_retorno: '), write(FimOuTemMais), nl, fail.

go :- tell('arctypes.txt'), findall(X, arcset(X),L), write(L),told.

```

## **Apêndice II. Propostas de Extensão à Aplicação de Empréstimo de Materiais**

Em anexo apresentamos os originais das propostas de extensão à aplicação de Empréstimo de Materiais. Lembramos que esses originais foram obtidos a partir da avaliação empírica descrita no Capítulo 5 e gentilmente cedidos pelos seus participantes. Os originais estão agrupados pelos diferentes participantes. O início de cada grupo de extensões é delimitado pelo formulário intitulado “Informações sobre a extensão sugerida” seguido de um número seqüencial de 1 a 6, indicador do respectivo participante (já que o anonimato dos mesmos foi garantido). A documentação do primeiro grupo de extensões aparece completa, inclusive com as folhas que não foram utilizadas pelo participante. Já os grupos seguintes mostram apenas as folhas que foram utilizadas para documentação de extensões.

O primeiro grupo de extensões é aquele em que o participante não delimitou claramente os limites entre uma extensão e outra. Nesse caso, a avaliadora foi quem estabeleceu os limites entre as extensões, escrevendo um número seqüencial ao lado de cada extensão (e.g. 1<sup>a</sup>, 2<sup>a</sup>, etc.).