Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Informática

Jair Cavalcanti Leite

Modelos e Formalismos para a Engenharia Semiótica de Interfaces de Usuário

Tese apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos necessários à obtenção do grau de Doutor em Ciências em Informática

Orientadora: Clarisse Sieckenius de Souza

Rio de Janeiro, 07 de Outubro de 1998

Resumo

Um dos requisitos para a usabilidade de um sistema interativo é que os usuários adquiram o conhecimento, denominado de modelo de usabilidade, sobre como aplicar as soluções-em-potencial concebidas pelo designer às tarefas do seu domínio. A abordagem da Engenharia Semiótica apresenta uma perspectiva na qual um sistema interativo é um artefato de metacomunicação através do qual o designer envia uma mensagem que comunica o modelo de usabilidade para o usuário. Partindo desta perspectiva e baseado nos conceitos de semiótica das teorias de Charles S. Peirce e de Umberto Eco desenvolvemos modelos teóricos que descrevem o modelo de usabilidade como sendo o conteúdo da mensagem do designer, a interface de usuário como a sua expressão e o design como sendo uma atividade de produção de signos apoiada por um sistema semiótico, composto por uma linguagem de especificação e por regras que correlacionam as mensagens especificadas aos widgets dos principais padrões e ferramentas de interfaces de usuário.

Abstract

Interactive systems usability could be enhanced if users learn all the knowledge - the usabilty model - that capability them in applying designer's potential solutions to domain tasks. The Semiotic Engineering approach perceives interactive systems as metacommunication artifacts that send a message from designer to users whose expression is the lower-level messages exchanged between user and system and whose content is the usability model. Starting from this perspective and based on semiotic theory concepts from Charles S. Peirce and Umberto Eco we present conceptual models to the interface as the expression and to the usability model as the content of designer's message. We also develop a semiotic system to support user interface design. The system is composed by a specification language and rules that maps specified messagens to user interface widgets. Our emphasis here is not in aesthetics aspects of user interfaces, but in the interactive and performing nature of the interface message as it is carried throughout the computational medium.

Índice Analítico

Índice de Figuras

Índice de Tabelas

Agradecimentos

À Thaís, a quem eu dedico este trabalho, que sempre incentivou este trabalho, me deu todo o apoio para que eu pudesse realizá-lo, soube compreender as dificuldades e teve muita, mas muita paciência.

À Clarisse, a quem eu também dedico este trabalho pela oportunidade de realizá-lo, pela parceria, pelo incentivo à qualidade e, principalmente, por ter sempre acreditado na minha capacidade.

Aos meus familiares, especialmente meus pais, Ernani e Marilena, e minha sogra, D. Lygia, pelo incentivo e apoio que precisei.

À Raquel que esteve sempre junto à mim, mesmo que a milhares de quilômetros, nas várias revisões, sugestões e discussões.

Aos colegas do SERG que muito me ajudaram a entender a Engenharia Semiótica e pelas sugestões valiosas que recebi.

À CAPES, ao CNPq, à VRAc da PUC-Rio e à UFRN pelo apoio financeiro ao longo de todos estes anos

Aos professores, funcionários e colegas da PUC-Rio que após tantos anos de trabalho tornaram-se grandes amigos. Um agradecimento especial ao Prof. Lucena e ao Prof. Hugo Fuks pelo grande incentivo no incio do doutorado.

Aos colegas professores da UFRN que abriram espaço para a realização deste trabalho.

À Larissa que nunca entendeu porque desde que ela nasceu eu estou “sempre neste computador”. Ela tem cinco anos.

INTRODUÇÃO

A utilização de sistemas computacionais como uma ferramenta intelectual tem sido discutida há mais de meio século. Apresentar teorias para instrumentar o processo de concepção de sistemas computacionais com ênfase na sua utilização por seres humanos é um desafio árduo cujo escopo é bastante abrangente. Várias abordagens visam atender ao *desafio de usabilidade*[1] [Adler & Winograd 92] utilizando conceitos das mais diversas áreas do conhecimento na elaboração de teorias sobre o processo de interação entre pessoas e computadores e aplicando-os ao *design de interfaces de usuário*.

Nosso trabalho visa mostrar como aplicar alguns conceitos teóricos de semiótica no processo de design de interfaces de usuário, com o objetivo de desenvolver modelos teóricos e ferramentas que permitam melhorar alguns aspectos do processo de design e da usabilidade de um sistema computacional interativo. Neste capítulo, vamos introduzir as motivações, os objetivos, o escopo e a estrutura desta tese.

Ferramentas Intelectuais e Mídia

Desde a metade do século, antes mesmo da difusão do uso de computadores no meio comercial nos anos 60, cientistas têm argumentado que sistemas computacionais devem ser projetados como uma ferramenta intelectual que aumenta a capacidade humana. Bush propôs a idéia de um engenho para armazenamento e recuperação de conhecimento - uma *enciclopédia* geral da humanidade [Bush 45]. Tal idéia, posteriormente denominada *hipertexto* por Ted Nelson [Nelson 65] veio a ter uma implementação real apenas nos anos 90 através da *World-Wide Web* [Berners-Lee 91]. Engelbart, foi um dos pioneiros em apresentar projetos com esta perspectiva - o *Augment* [Engelbart 63]. Ele também foi um dos pioneiros a ressaltar a importância da interface de

usuário e desenvolver dispositivos para a sua implementação como forma de atingir este objetivo. McLuhan considera artefatos tecnológicos, sob a denominação de *mídia*, como a extensão das capacidades humanas [McLuhan 64]. Alan Kay, considerado o idealizador dos computadores pessoais, projetou o *Dynabook* como uma *mídia dinâmica para pensamento criativo* [Kay & Goldberg 77].

Um sistema computacional deve ser visto como um mídia intelectual *quente* no sentido proposto por McLuhan ou *ativo* como explicitamente foi designado por Alan Kay. Isto significa que esta mídia não deve ter apenas a capacidade de veicular informações e de reproduzir qualquer outra mídia existente, mas também de interagir com o usuário. Neste caso, ele é uma *"meta-medium" ativa* que pode responder a questões e experimentos, e possibilitar ao usuário aprender através do processo de interação com sistemas [Kay & Goldberg 77].

Adler e Winograd ressaltaram esta postura argumentando que novas tecnologias serão mais eficazes quando projetadas para aumentar, ao invés de substituir, as habilidades dos usuários [Adler & Winograd 92]. Eles denominam *desafio de usabilidade* o projeto de novas tecnologias que buscam explorar ao máximo as habilidades dos usuários na criação de ambientes de trabalho mais eficazes e produtivos. Este desafio abandona a tradicional perspectiva que considera o usuário como peça integrante do sistema - um autômato - na qual os únicos critérios de usabilidade são aqueles que visam minimizar o esforço físico dos usuários.

Norman, um dos mais influentes pesquisadores e um dos pioneiros na aplicação de psicologia e ciência cognitiva ao design de interfaces de usuário, recentemente tem enfatizado que a tecnologia deve ser projetada com o objetivo de ajudar as pessoas a serem mais espertas, eficientes e inteligentes. Ele enfatiza a necessidade da construção de *artefatos cognitivos* que *"tornam as pessoas mais espertas"* [Norman 91; Norman 93].

O problema tratado nesta tese, compartilha com estes trabalhos clássicos a perspectiva que considera que um sistema interativo deve ser projetado como ferramenta intelectual ou mídia de maneira a possibilitar aos usuários resolverem problemas e aprenderem através da interação.

---

[1]Do inglês *usability.*

Usabilidade

Para adequar-se aos pontos de vista mencionados na seção anterior, o desenvolvimento de sistemas computacionais interativos não pode ser considerado apenas dentro do escopo restrito da ciência da computação, da engenharia de sistemas e de software e de fatores humanos (ergonomia). As habilidades dos usuários, a situação de uso e o contexto onde eles estão envolvidos - o domínio de aplicação - são fundamentais para a usabilidade e devem ser considerados no desenvolvimento de sistemas interativos [Adler & Winograd 92].

A *usabilidade* é um conceito que se refere à qualidade da interação de sistemas com os usuários e depende de vários aspectos. Dentre eles, nos concentraremos na *facilidade de aprendizado*[2] do sistema que é o tempo e o esforço necessários para os usuários atingirem um determinado nível de desempenho [Shackel 89]. Em outras palavras, estamos interessados na facilidade de aquisição pelo usuário da competência necessária para interagir com o sistema.

A *competência do usuário* é o conhecimento a respeito do sistema necessário para ele desempenhar a interação com o sistema e é denominado de *modelo conceitual do usuário* [Moran 81; Norman 86b; Liddle 96]. Por ser um fator chave da usabilidade, denominá-lo-emos ***modelo conceitual de usabilidade***.

Norman considera que a maneira de se adquirir o modelo conceitual de usabilidade é através da interface e da documentação que acompanha o sistema, às quais ele denomina *imagem do sistema* [Norman 86b]. A imagem do sistema deve veicular o modelo conceitual de maneira mais clara e óbvia possível para que os usuários não desenvolvam um modelo conceitual incorreto [Preece 94]. Assim, tudo no sistema deve estar direcionado a tornar este modelo coerente e lógico uma vez que ele determina as atividades cognitivas que o usuário deve desempenhar para melhor utilizar o sistema [Liddle 96].

O modelo conceitual de usabilidade envolve tudo aquilo que o sistema oferece ao usuário - a sua *funcionalidade* - e a maneira como ele permite o usuário interagir - a sua *interatividade.* A funcionalidade é determinada pelo *modelo funcional da aplicação* que

[2]Do inglês *Learnability.*

visa descrever quais as funções são oferecidas ao usuário como solução para os seus problemas. Ela deve estar adequada às tarefas dos usuários, inserida no seu domínio de aplicação e por ele motivada, e deve poder ser estendida pelo próprio usuário. A interatividade é determinada pelo *modelo de interação* que visa descrever os modos (regras e protocolos) pelos quais os usuários interagem com o sistema. Ela deve estar adequada as capacidades físicas e cognitivas dos usuários.

O desafio de usabilidade requer que o usuário tire proveito de todo o potencial de um sistema que tenha sido desenvolvido como uma ferramenta intelectual. Por outro lado, num processo simbiótico [Licklider 60], a ferramenta intelectual deve potencializar maior usabilidade aumentando a competência do usuário.

Nosso enfoque para o desafio de usabilidade está no aumento da competência do usuário através do desenvolvimento de sistemas que não apenas sejam fáceis de aprender e de usar, mas que apóiem a aquisição do modelo de interação e de funcionalidade do sistema.

Design de Interfaces de Usuário

O desenvolvimento de sistemas computacionais interativos é uma atividade bastante complexa que abrange diversas etapas e requer o envolvimento de pessoas com conhecimento em diversas áreas. Dentre estas etapas, o design do sistema envolve as atividades de concepção dos seus diversos componentes e a especificação de seus modelos.

No nosso enfoque para o desafio de usabilidade, o design do sistema está restrito à concepção e especificação dos componentes conceituais do modelo de funcionalidade e do modelo de interação que serão disponibilizados para os usuários. O modelo conceitual de usabilidade do designer diz respeito às funções do sistema e aos modos de interação que ele achou necessário para o usuário realizar as suas tarefas. Durante o processo de interação com o sistema o usuário precisa recorrer ao seu modelo conceitual de usabilidade para poder desempenhar as ações que acionam as funções do sistema necessárias para atingir suas metas. A Figura 1-1 ilustra estes dois aspectos.

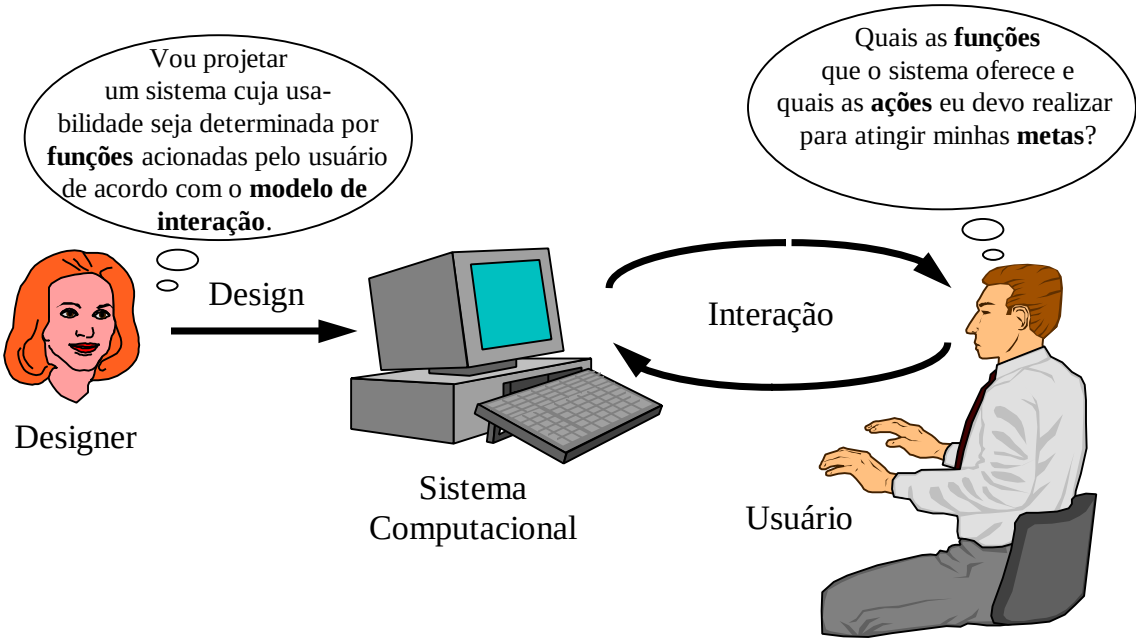


Figura 1-1:O design de sistemas e a interação usuário-sistema

A *interface de usuário* é a parte do artefato de software com a qual o usuário entra em contato - física, perceptiva e cognitivamente - na realização de tarefas no seu domínio de atividades [Moran 81]. Ela é composta por uma coleção de dispositivos através dos quais o usuário pode trocar informações com o sistema. A troca de informações ocorre através de estruturas de interações tais como menus, janelas, ícones, linguagens de comandos, formulários, perguntas e respostas em linguagem natural, dentre outras que determinam o *modelo de interação.*

O *design de interface de usuário* é o processo de concepção dos objetos de software e hardware que determinam os modos e as estruturas de interação - o modelo de interação.

Além da função de interação usuário-sistema, a interface pode desempenhar um outro papel que é importantíssimo para a aquisição do modelo conceitual de usabilidade do usuário. Ela pode ser o mecanismo de comunicação que veicula e ensina o modelo de usabilidade para o usuário. Para colocarmos em prática esta perspectiva é necessário enfatizarmos o papel do projetista como agente idealizador de um processo metacomunicativo através do qual o conhecimento será veiculado pela mídia

computacional. O design de interfaces deve ser estendido de maneira que elas possam também ser concebidas com o objetivo de ensinar o modelo de usabilidade.

A Necessidade de Fundamentação Teórica

O problema que estamos tentando resolver envolve oferecer algum conhecimento teórico, modelos e formalismos para que o design de interfaces de usuário se aproxime de seu propósito de design para usabilidade. Na indústria de software, o design de interface de usuários tem sido conduzido através de processos iterativos de construção e avaliação de protótipos baseados em princípios e diretrizes empíricas, tal como proposto em *The Windows Interface: Guidelines for Software Design* [Microsoft 95] and *Macintosh Human Interface Guidelines* [Apple 92]. No processo de desenvolvimento é importante basear a prática de design em uma fundamentação teórica, pois modelos obtidos da aplicação de teorias possibilitam explicações e predições para fenômenos de interação entre usuários e sistemas.

A área de Interação entre Seres Humanos e Sistemas Computacionais[3] (IHC) tem por objetivo principal fornecer aos pesquisadores e desenvolvedores de sistemas *explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário* [ACM SIGCHI 92]. Com teorias a respeito dos fenômenos envolvidos seria possível prever antecipadamente se o sistema a ser desenvolvido satisfaz as necessidades de usabilidade dos usuários. IHC busca desenvolver modelos teóricos de desempenho e cognição humanos, bem como técnicas efetivas para avaliar a usabilidade [Lindgaard 94].

O nosso trabalho aplica conhecimento teórico de *semiótica* na elaboração de modelos que oferecem explicações para fenômenos relacionados ao design de interface. Baseado neste modelo, elaboramos formalismos que auxiliam o processo de design de interfaces com o objetivo de facilitar o aprendizado do modelo de usabilidade.

A Engenharia Semiótica

Diversas abordagens teóricas têm apresentado propostas para o processo de design para usabilidade, como será visto no capítulo 3. Entretanto, a maioria destas abordagens

[3]Esta é a tradução que propomos para o termo em inglês tradicionalmente utilizado na literatura: *Human-Computer Interaction*. O acrônimo IHC vem justamente da tradução literal, para o português, do termo em

apresentam como solução a este desafio modelos e técnicas para a concepção do modelo de usabilidade a partir de características cognitivas dos usuários. O objetivo é sempre um melhor design conhecendo-se um perfil cognitivo dos usuários. Embora se tenha quase sempre conseguido atingir este objetivo, ele é insuficiente para o desafio de usabilidade por não articular o papel do projetista e da interface na aquisição do modelo de usabilidade pelo usuário. Torna-se necessário uma nova abordagem teórica para o design de interfaces de usuário.

A Engenharia Semiótica é uma abordagem baseada em teoria para o design de interfaces de usuário na qual interfaces são consideradas como artefatos de *metacomunicação* [de Souza 93]. Isto significa que interfaces comunicam mensagens enviadas dos designers para os usuários e estas mensagens, por sua vez, enviam e recebem mensagens nos intercâmbios com os usuários. A mensagem designer-usuário deve ter como significado a resposta a duas perguntas fundamentais: (I) "Quais os tipos de problemas que este sistema está preparado para resolver?", i.e., qual é a *funcionalidade do sistema*; e (II) "Como estes problemas podem ser resolvidos?", i.e., qual é o *modelo de interação.* A Engenharia Semiótica especializa abordagens semióticas concentrando-se na estrutura e expressividade de mensagens que se pode enviar e receber através de códigos computáveis.

Esta proposta está de acordo com a visão de "meta-medium" ativo, discutida anteriormente [Kay & Goldberg 77], mas foi idealizada a partir da perspectiva de mídia proposta em [Kammersgaard 88]. Kammersgaard propôs que, dentre outras perspectivas, sistemas computacionais podem ser vistos como um *medium* de comunicação. Existem várias maneiras para se utilizarem os aspectos comunicativos de computadores. Uma delas é a comunicação unidirecional de designers para usuários que acontece quando um sistema projetado por uma pessoa é utilizado por outra. Andersen considera o designer como tendo o papel de um emissor num conjunto de emissores, cada qual contribuindo com cada mensagem que é enviada através do medium (P. B. Andersen 85, citado em [Kammersgaard 88]).

---

inglês. Outros termos como interação pessoa-computador e interação usuário-sistema são termos sinônimos usados no mesmo sentido do termo que adotamos aqui.

Em seu escopo mais amplo, a Engenharia Semiótica visa apresentar condições teóricas e técnicas para o desenvolvimento de *interfaces e linguagens para usuários* que consideram sistemas como ferramenta intelectual, visando responder ao desafio de usabilidade. O aspecto de usabilidade que a Engenharia Semiótica visa resolver é como o conhecimento que o usuário precisa adquirir pode ser melhor "ensinado" através da interface de usuário, e quais aspectos da interface possibilitam um melhor interpretação do modelo de interação e de funcionalidade (o modelo de usabilidade) do sistema. Nosso objetivo específico é oferecer mecanismos de design que possibilitem a produção de interfaces que permitam ao usuário entender que um determinado sistema foi projetado por uma pessoa visando soluções que ela acha necessárias e adequadas para os problemas do usuário.

A Engenharia Semiótica, tal com outras abordagens [Andersen 90, Nadin 88], elege a *semiótica* como fonte de conhecimento teórico para IHC. Semiótica é a disciplina que investiga comunicação e cultura. Cultura e conhecimento são manifestados e compartilhados numa sociedade através de *signos*. Signo é o termo mais preciso para se referir a algo que representa uma outra coisa para alguém. Ele abrange símbolos, ícones, palavras, sinais, índices (indícios), dentre outros que podem ser utilizados em qualquer forma de comunicação ou aquisição de conhecimento.

A semiótica proporciona uma perspectiva pragmática para design como uma atividade metacomunicativa. Esta perspectiva social e pragmática é também manifestada em [Adler & Winograd 92; Winograd 96; Rheinfrank & Evenson 96] que introduzem, respectivamente, significados particulares para os conhecidos conceitos de *usabilidade*, *design de software e linguagens de design*. Nosso trabalho segue estes conceitos, colocando-os dentro de uma perspectiva semiótica.

Objetivos

Seguindo a abordagem da Engenharia Semiótica, o nosso trabalho aplica teoria semiótica com o objetivo de interpretar problemas tradicionais através de uma nova *perspectiva*, elaborar um *modelo teórico* que oferece explicações e predições para alguns aspectos do processo de design e da interação usuário-sistema e desenvolver um formalismo que possibilite a aplicação do modelo teórico na prática de design.

Nosso modelo parte de uma perspectiva teórica fundamentada na definição de *signo* proposta por Peirce [Peirce 31], na *Teoria dos Códigos e da Produção de Signos* de Eco [Eco 76] e na *Engenharia Semiótica* [de Souza 93]. O ponto principal do trabalho é a hipótese de que o modelo de usabilidade pode ser adquirido durante o processo de interação pela interpretação das mensagens da interface de usuário.

Os conceitos de semiótica são fundamentais na elaboração de modelos téoricos para sustentar os formalismos e ferramentas que orientam o design. Os modelos que propomos neste trabalho são fundamentais determinar como um *sistema semiótico de apoio ao design* pode vir a ser construído. O sistema semiótico que propomos abstrai os objetos de interfaces (*widgets*) através do conceito de *signos de interface* que devem ser articulados com os elementos de uma linguagem de especificação do modelo de usabilidade. Ele é, portanto, o formalismo que aplica o modelo teórico na prática de design.

O objetivo deste sistema semiótico é apoiar a elaboração da mensagem do designer a ser veiculada através do sistema, oferecendo recursos para a elaboração da sua expressão e do seu conteúdo. Ele deve orientar o designer na escolha dos elementos expressivos da interface de usuário (os widgets, por exemplo) que permitam ao usuário intepretar as funções, os objetos e os modos de interação com o sistema. Desta maneira, ele contribui na facilidade de aprendizado do sistema através de uma interface que comunica o modelo de usabilidade que o designer concebeu.

Estrutura da Tese

Para descrever como os objetivos delineados acima foram atingidos, estruturamos este trabalho da seguinte forma.

O capítulo 2 descreve o conhecimento básico necessário para a compreensão desta tese. A área de IHC é introduzida e discutida. Conceitos relativos à interface de usuário, ao processo de interação usuário-sistema e ao design de interfaces são também introduzidos. O escopo e o foco do nosso trabalho são identificados em relação ao processo global de desenvolvimento de sistemas computacionais.

O capítulo 3 apresenta e discute diversas abordagens teóricas para o design de interfaces, categorizando-as em abordagens cognitivas e semióticas, apresentando suas similaridades e diferenças, vantagens e desvantagens.

Em seguida, no capítulo 4, apresentamos a abordagem original da Engenharia Semiótica a partir da qual desenvolvemos o nosso trabalho, exemplificando-a através de interfaces gráficas e expandindo a sua perspectiva original de maneira a revelar como problemas do desenvolvimento de software podem ser reinterpretados. O ponto fundamental é que, baseado no conceito de *signo* de Peirce [Peirce 31], o processo de interação usuário-sistema pode ser visto como um signo produzido pelo designer (a *mensagem do designer*). A conclusão obtida é que o seu design é um processo de produção de signo que deve ser fundamentado por conceitos de semiótica.

O capítulo 5 apresenta os fundamentos, requisitos e os modelos teóricos necessários para o design do sistema como produção de signo. A partir da *Teoria de Produção de Signos* apresentamos os modelos téoricos nos quais a interface é a expressão e o modelo de usabilidade é o conteúdo da mensagem do designer. Estes modelos são fundamentais para a elaboração do formalismo de apoio à produção do signo.

O capítulo 6 apresenta um sistema semiótico de apoio ao design de interfaces de usuário. Como um formalismo de produção de signo ele requer um sistema para a especificação do conteúdo que possa ser correlacionado com um sistema para a especificação da expressão. Baseados nos modelos teóricos apresentado no capítulo 5, elaboramos uma linguagem para a especificação do conteúdo da mensagem do designer. Descrevemos ainda como os principais padrões e ferramentas de interfaces existentes podem funcionar como sistema para a construção da expressão quando correlacionados à linguagem de especificação.

Por fim, no capítulo 7, apresentamos um sumário das principais contribuições apresentadas, discutindo suas principais implicações. Fazemos algumas comparações com as abordagens cognitivas e semióticas e descrevemos as principais extensões necessárias a serem feitas em trabalhos futuros.

Escopo e Limitações

Embora nosso tema esteja relacionado com o desenvolvimento de sistemas computacionais, o escopo deste trabalho abrange os aspectos conceituais do design de interfaces de usuário e da interação entre usuários e sistemas. Mais especificamente, nosso foco está em saber como podemos melhorar o processo de design de maneira que os sistemas produzidos possam comunicar mais facilmente o conhecimento necessário a sua utilização. Apesar de importantes, por questões de tratabilidade e delimitação temática, não são considerados outras etapas do processo de desenvolvimento tais como a análise de requisitos, a implementação e a sua avaliação. Mesmo no escopo de interfaces de usuário, também não são tratados aspectos de design e implementação de dispositivos de hardware ou de arquiteturas, algoritmos e modelos de dados do software que implementa a interface.

O nosso foco para usabilidade está restrito ao aspecto de *facilidade de aprendizado*, a qualidade que o sistema tem de ser *fácil de aprender.* As abordagens mais comuns para design centrado-no-usuário buscam resolver este problema através do estudo das capacidades e limitações cognitivas dos usuários e, com isto, elaborar modelos que permitam prever e analisar quais características os sistemas, e em especial as suas interfaces, devem ter de maneira a serem mais fáceis de aprender. A nossa abordagem é complementar, uma vez que ela visa facilitar o aprendizado do usuário através da elaboração de mensagens que *ensinam* aquilo que é necessário para uma melhor utilização do sistema. Nosso trabalho complementa o foco tradicional na facilidade de aprendizado (*Learnability*) com o foco no potencial de expressividade e comunicação que pode ser oferecido ao designer para *ensinar* o modelo de usabilidade ao usuário (*Teachability*) [de Souza & Umiker-Sebeok 98].

Limitado a este enfoque para a usabilidade, nosso trabalho deve ser avaliado apenas pelo apoio que ele oferece a este processo comunicativo associado ao design da interface de usuário. O apoio é dado através de modelos teóricos e formalismos que instrumentam o processo e devem ser integrados a outras ferramentas e etapas do processo de design do sistema.

Pelos aspectos mencionados acima, nossos modelos e formalismos por si só não garantem que as interfaces produzidas com o seu apoio sejam mais fáceis de aprender e de usar. Elas precisam ser complementadas por abordagens que especifiquem qual a funcionalidade necessária e qual o modelo de interação mais adequado para o usuário de acordo com suas capacidades e limitações físicas e cognitivas. O primeiro aspecto é abordado pela engenharia de sistemas e de software, enquanto o segundo é do escopo das abordagens cognitivas para o design de sistemas centrado-no-usuário.

Estaremos limitados a interfaces gráficas de usuário (GUI) no estilo *WIMP* (do inglês: Windows-Janelas, Icons-Ícones, Menus e Pointers-Apontadores). Também estaremos concentrados primordialmente em aplicações de software destinadas a computadores pessoais, com usuário único. Uma GUI se caracteriza por ser uma interface virtual *"on-screen"* onde dispositivos de interação - os *widgets* - são apresentados na tela para serem acionados com o mouse ou outro dispositivo equivalente. O hardware básico desta interface é constituído pela tela, pelo teclado e pelo mouse.

Outros trabalhos em desenvolvimento pelo grupo *SERG* (Semiotic Engineering Research Group), da PUC-Rio, complementam estas limitações. A Engenharia Semiótica de interfaces de usuário com manipulação direta é considerada em [Martins 98]. Uma abordagem semiótica para interfaces multi-usuários está sendo desenvolvida, como mostrado em [Prates 97]. Extensões da interface para programação de usuário final dentro da perspectiva da Engenharia Semiótica foi abordado em [de Souza 96] e [de Souza 97]. Por fim, linguagens de interface para aquisição de conhecimento diretamente do usuário para sistemas baseados em conhecimento são ainda objetivos da Engenharia Semiótica em [de Souza 97] e [de Souza e Umiker-Sebeok 98].

CONCEITOS BÁSICOS

O desafio de usabilidade é um problema bastante genérico e amplo para que possamos resolvê-lo por completo. O processo de interação entre sistemas computacionais e seres humanos, por envolver participantes de natureza distinta, requer um estudo multidisciplinar que elabore a instrumentação necessária ao desenvolvimento dos sistemas. O ambiente de utilização também exerce bastante influência na usabilidade e, como tal, também deve ser considerado por pesquisadores e desenvolvedores. Mais do que desenvolver um sistema, o desafio de usabilidade requer o desenvolvimento da sua aplicação integrada ao ambiente do usuário de maneira a estender sua capacidade.

Neste capítulo apresentaremos o contexto no qual o tema deste trabalho está inserido, introduzindo a área de pesquisa e as disciplinas relacionadas, o escopo da nossa abordagem, os conceitos básicos necessários à sua compreensão, em qual etapa do desenvolvimento de sistemas está o nosso foco e qual o tipo de instrumentação que pode ser utilizada.

A Área de Interação entre Seres Humanos e Sistemas Computacionais

O estudo de como os sentidos e as capacidades motoras permitem às pessoas utilizarem máquinas e ferramentas complexas já era objeto de estudo da disciplina *Fatores humanos*, como é chamada nos E.U.A, ou *Ergonomia*, na Europa, durante a segunda guerra mundial. Estas disciplinas, historicamente, abordam o desempenho das pessoas no uso de máquinas em geral. Como computadores são máquinas destinadas ao processamento e uso de informações, tornou-se necessário estender esta disciplina para se estudar também a capacidade mental que possibilita às pessoas produzirem informações processáveis por computadores, recuperá-las e compreendê-las. Esta extensão surgiu nos anos 80 com rótulo de Interação entre Seres Humanos e Sistema Computacionais (IHC).

De acordo com a definição apresentada na introdução, IHC é a disciplina que visa fornecer aos pesquisadores e desenvolvedores de sistemas *explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário* [ACM SIGCHI 92].

Disciplinas envolvidas

Na condição de área ou campo de pesquisa, IHC deveria apresentar teorias e modelos a respeito dos elementos e processos de design e de interação envolvidos. Alguns autores, entretanto, consideram-na apenas como a reunião de estudos inter-disciplinares, não constituindo em si uma disciplina científica [Dix et al. 93, Booth 88, Lindgaard 94]. Dix argumenta que não se terá uma teoria unificada de IHC devido à complexidade e à natureza dos elementos envolvidos.

IHC deve envolver não apenas ergonomia, análise de sistemas e engenharia de software, disciplinas mais tradicionalmente associadas ao desenvolvimento de sistemas computacionais, mas também disciplinas que estudem o comportamento e a capacidade humana, tais como as disciplinas da psicologia e ciências cognitivas, bem como disciplinas que abordem os fatores sociais deste comportamento.

IHC, portanto, é multi-disciplinar e está relacionada com áreas tais como: *ciência da computação,* que sustenta o conhecimento tecnológico para a engenharia de software e hardware; *ergonomia,* para o estudo do desempenho físico do usuário; *psicologia* e *ciência cognitiva,* que oferecem conhecimento sobre comportamento e habilidades perceptivas e cognitivas, bem como técnicas de análise e avaliação empírica; *sociologia,* para ajudar o designer a entender o contexto da interação; *design gráfico,* para a produção dos elementos gráficos apresentados; *marketing,* para a sua comercialização; *antropologia,* para o estudo de características humanas; e várias outras [Dix et al. 93]. Preece acrescenta ainda *engenharia, lingüística e filosofia* [Preece 94].

A característica multi-disciplinar abre espaço para se envolver novas disciplinas. O nosso trabalho aborda alguns dos problemas de IHC por uma nova perspectiva, através da aplicação de uma disciplina téorica, não tradicionalmente incluída: a semiótica[4]. A

[4] *Semiótica* é a disciplina que estuda os signos, sistemas de significação e sistemas de comunicação [Eco 88]. Uma definição equivalente já foi introduzida no capítulo 1. Maiores detalhes serão mostrados no capítulo 3.

semiótica tem sido mais uma das ciências humanas a trazer fundamentação teórica para a IHC, como demonstrado nos trabalhos de [Nadin 88, Andersen 90, de Souza 93 e Nake 94]. O foco da semiótica está na cultura e nos sistemas de signos que apóiam as atividades cognitivas humanas. Alguns conceitos de semiótica, necessários à compreensão do nosso trabalho, serão apresentados mais adiante.

Diferentes enfoques da IHC

Um dos objetivos dos pesquisadores envolvidos com IHC foi chamar a atenção dos desenvolvedores para enfocar não apenas o sistema e sua interface, mas também os usuários envolvidos e o próprio processo de interação do qual eles participam. A esta proposta de mudança do foco deu-se o nome de *Design de Sistemas Centrado-no-Usuário* [Norman & Draper 86]. Nesta abordagem, o usuário e o sistema são identificados como os dois focos de interesses principais. A ciência da computação oferece o conhecimento para o desenvolvimento do sistema. As ciências humanas permitem abordagens que possibilitam um design de sistemas *centrado-no-usuário.*

No contexto de IHC devemos considerar quatro elementos básicos: *o sistema, os usuários, os desenvolvedores e o ambiente de uso* (domínio de aplicação) [Dix et al. 93]. Estes elementos estão envolvidos em dois processos importantes: a *interação usuário-sistema* e o *desenvolvimento do sistema.* O *curriculum* proposto para IHC identifica cinco enfoques para o estudo destes elementos e para a sua aplicação na melhoria dos processos de desenvolvimento e de interação usuário-sistema. Para cada um destes focos, diferentes disciplinas proporcionam os estudos teóricos que podem ser aplicados ao desenvolvimento. Este esquema pode ser visualizado na Figura 0-1.

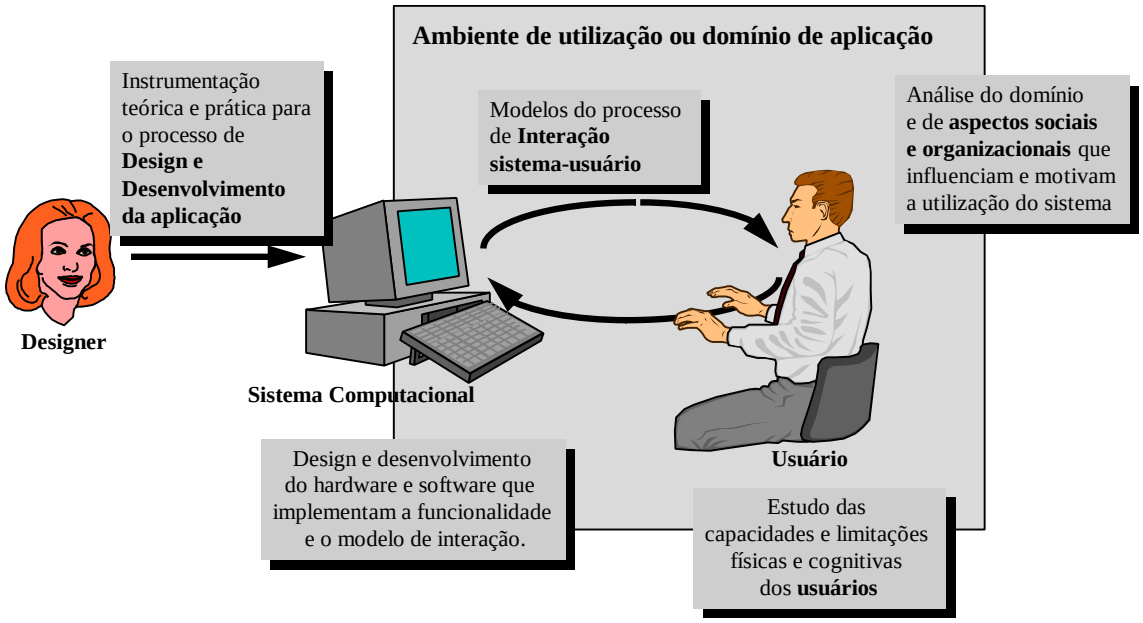


Figura 0-1: Diferentes enfoques da IHC

Casaday, numa postura mais prática voltada para desenvolvedores, considera que a IHC pode ser esquematizada de maneira a revelar seis focos de atividades onde decisões de design devem ser tomadas de forma *balanceada* [Casaday 91]. Estes focos são: *contexto para design, atividades mentais dos usuários, design do modelo funcional da interface, atividades físicas do usuário, design físico da interface* e *implementação.* Neste modelo, embora o autor não mencione explicitamente, pode-se identificar claramente duas distinções categóricas importantes.

A primeira diz respeito às *atividades de interação* entre o usuário e o sistema - as atividades mentais e físicas dos usuários, bem como o contexto onde elas ocorrem - e às *atividades de design e desenvolvimento*. A distinção entre a interação e o desenvolvimento também é abordada em [Hix & Hartson 93] como pertencentes a dois domínios: o *comportamental*, onde ocorrem as atividades de interação como o usuário; e o *construcional*, que diz respeito ao desenvolvimento do software da interface.

A outra distinção importante diz respeito aos elementos *conceituais* e *físicos* que são ortogonais aos primeiros. O contexto, as atividades mentais e o modelo funcional da interface dizem respeito a componentes conceituais enquanto que o atividades físicas do usuário, design físico da interface e implementação pertencem a um domínio físico.

Embora o conceitual seja dependente do físico, eles podem ser estudados independentemente.

Estas distinções evidenciam que é possível e aconselhável separar os aspectos conceituais dos físicos. O design da interface de usuário e da funcionalidade é a parte do desenvolvimento pertencente a um domínio conceitual e não precisa abordar aspectos de implementação de software e hardware. O nosso trabalho está restrito aos fenômenos do domínio conceitual.

O escopo do design no processo de desenvolvimento

O processo de desenvolvimento de sistemas computacionais ainda é alvo de diversas pesquisas. Existem diferentes metodologias que apresentam os mais diversos modelos do processo, com diferentes etapas e abordagens para cada uma delas e apoiados por diferentes formas de instrumentação.

O ciclo de vida clássico de um sistema descreve um modelo *em cascata* para as etapas do seu desenvolvimento. Embora seja alvo de diversas críticas por especialistas da área, este modelo ilustra de forma clara a clássica distinção entre as etapas *análise, design, implementação e avaliação* [Pressman 96]. O modelo *em espiral* ascrescenta a estas etapas a *definição, o planejamento e a análise de riscos* a serem verificadas periodicamente num processo evolutivo.

A análise tem por objetivo o levantamento dos elementos do domínio de aplicação e deve gerar como produto uma especificação dos requisitos aos quais o sistema deve atender. Esta atividade é apoiada por técnicas de comunicação com clientes e usuários, e por notações ou linguagens que permitem descrever todos os elementos do domínio de maneira abstrata.

O design envolve a concepção das soluções que atendem aos requisitos de usuários, a sua especificação através de linguagens apropriadas, a avaliação de diferentes alternativas de soluções e a *tomada-de-decisão* a respeito de cada um delas. Tais soluções são os vários componentes e processos A concepção, por ser uma atividade mental que envolve criatividade, é bastante difícil de ser conduzida. Não se sabe exatamente como nem quais as soluções a serem concebidas. Como o nosso enfoque está nesta atividade,

na seção seguinte vamos mostrar, resumidamente, os elementos do sistema a serem concebidos e algumas formas de instrumentação que apóiam esta atividade.

Na implementação as soluções concebidas e especificadas devem ser concretizadas em elementos de hardware e software do sistema. A implementação do software é feita através da codificação em uma linguagem de programação a ser intepretada ou compilada e executada pelo hardware.

Estas atividades não são etapas completamente independentes e dissociadas de um processo de desenvolvimento. Modelos alternativos do processo enfatizam na estreita ligação entre análise e design [Wasserman 96]. A investigação dos problemas e a apresentação das suas soluções são ambas atividades mentais que nunca são completamente independentes. Os chamados modelos da análise, um Diagrama de Fluxo de Dados por exemplo, muitas vezes apresentam decisões que são na verdade soluções dadas pelos analistas. A avaliação não deve ser realizada apenas ao final do processo de desenvolvimento, mas ao longo de todo o processo. A análise, o design e a implementação devem todos ser avaliados de maneira adequada.

Recentemente, vários autores com as mais variadas formações e experiências, liderados por Winograd, apresentaram propostas que caracterizam a prática, a profissão e a disciplina de *Design de Software* [Winograd 96]. O sentido deste termo é o de concepção da aplicação e não do projeto da arquitetura, estruturas e algoritmos de um software ou programa. Desta forma, a atividade de design da aplicação e da interface não deve ser confundida com a atividade de especificação da maneira proposta em engenharia de software, nem com o processo de construção do software. Design de software é o ato de determinar a experiência do usuário com uma aplicação de software [Liddle 96], e deve ser independente do funcionamento e da implementação do programa que faz a aplicação do software trabalhar.

O nosso enfoque para o design de software está limitado ao domínio conceitual e abrange o *design da interface de usuário,* do *modelo de interação* e da *funcionalidade*, que chamamos de *modelo de usabilidade* da aplicação. Todos estes conceitos serão discutidos na próxima seção.

O termo ***sistema computacional*** refere-se aos artefatos eletrônicos digitais (hardware) que adquirem *funcionalidade* através de programas que processam estruturas dados e instruções (software). No desenvolvimento de um sistema[5] ambos precisam ser especificados. Entretanto, o hadrware é uma máquina de propósito geral e a sua funcionalidade é determinada pelo software. Devido a esta característica as atenções ficam quase todas voltadas para o desenvolvimento do software.

Um ***sistema interativo*** permite que o seu funcionamento possa ser controlado pelos seus usuários. Num sistema interativo é importante distinguir *a interface de usuário*, que determina um modelo de interação, do seu *núcleo funcional*, que determina a sua *funcionalidade.*

Aplicação de software

O termo ***aplicação de software*** refere-se ao software como um produto aplicado ao domínio de utilização. Ele deve ser interpretado no sentido de *programa executando,* i.e., sendo utilizado pelos usuários com o objetivo de solucionar seus problemas. Este termo é conceitualmente diferente de *programa fonte* que se refere à descrição estática em uma linguagem de programação. Um *programa fonte* descreve as estruturas de uma aplicação de software. Para se referir ao programa fonte utilizamos o termo *arquitetura do software* ou *estrutura do software.*

A *aplicação de software* pode ser vista como uma *virtualidade*, ou seja, uma máquina virtual destinada a resolver problemas específicos de seus usuários [Winograd 96]. São máquinas que operam sobre o domínio conceitual de informação e conhecimento. Uma aplicação de software, como entidade virtual, existe apenas em um sistema computacional. O hardware funciona como um meio que possibilita a existência desta máquina simbólica.

Os termos aplicação de software, aplicação e software são usados como sinônimos, uma vez que um sistema computacional interativo apenas tem utilidade

[5]Ao longo deste trabalho quando usarmos a palavra *sistema* estamos nos referindo a sistema computacional interativo.

quando o software lhe dá uma *funcionalidade* que pode ser aplicado ao domínio do usuário.

Funcionalidade

O termo *funcionalidade* designa o aspecto do sistema computacional que retrata as funções necessárias para a resolução de problemas em um domínio específico. A funcionalidade se refere àquilo que um programa faz e, no caso de software interativo, o que ele deve oferecer para seus usuários. Funcionalidade é diferente de *funcionamento* que se refere a como um software realiza as suas funções. A estrutura do software (seus tipos de dados e algoritmos) do programa fonte determina este funcionamento.

A funcionalidade é determinada pelo *modelo funcional da aplicação,* também chamado de modelo de funcionalidade. Na engenharia de software, o modelo de funcionalidade é elaborado na fase de *especificação funcional de software* a partir das informações obtidas durante a *análise de requisitos.*

Interface de Usuário

O termo *interface* é aplicado normalmente àquilo que interliga dois sistemas. Tradicionalmente, considera-se que uma interface homem-máquina é a parte de um artefato que permite a um usuário controlar e avaliar o funcionamento do mesmo através de dispositivos sensíveis às suas ações e capazes de estimular sua percepção. No processo de *interação* usuário-sistema a interface é o combinado de software e hardware necessário para viabilizar e facilitar os processos de comunicação entre o usuário e a aplicação. A interface entre usuários e sistemas computacionais diferencia-se das interfaces de máquinas convencionais por exigir dos usuários um maior esforço cognitivo em atividades de interpretação e expressão das informações que o sistema processa [Norman 86].

Segundo Moran, *a interface de usuário deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato física, perceptiva e conceitualmente* [Moran 81]. Esta definição de Moran caracteriza uma perspectiva para a interface de usuário como tendo um componente físico, que o usuário percebe e manipula, e outro conceitual, que o usuário interpreta, processa e raciocina. Moran e outros denominam este componente de *modelo conceitual do usuário.*

A interface é tanto um *meio* para a interação usuário-sistema, quanto uma *ferramenta* que oferece os instrumentos para este processo comunicativo. Desta forma a interface é um *sistema de comunicação.* Quando se considera a aplicação como máquina(s) virtual(is), a interface pode ser considerada ainda como um *ambiente virtual para ações.*

A interface possui componentes de software e hardware. Os componentes de hardware compreendem os dispositivos com os quais os usuários realizam as atividades motoras e perceptivas. Entre eles estão a tela, o teclado, o mouse e vários outros. O *software da interface* é a parte do sistema que implementa os processos computacionais necessários para controle dos dispositivos de hardware, para a construção dos dispositivos virtuais (os *widgets*) com os quais o usuário também pode interagir, para a geração dos diversos símbolos e mensagens que representam as informações do sistema e para a interpretação dos comandos dos usuários.

Modelo de interação

Interação é o processo de comunicação que ocorre entre um usuário e uma aplicação de software. Não será usado aqui o termo *diálogo*, nem tampouco *conversação*, por acharmos que essas palavras se empregam mais adequadamente quando a interação é uma comunicação entre dois agentes de natureza cognitiva semelhante, isto é, que possuem conhecimento, tradição cultural e linguagem comuns.

O ***modelo de interação*** é o conjunto de protocolos que permite ao usuário interagir com a aplicação. Estes protocolos muitas vezes são determinados por uma linguagem, e por isso é também chamado de *linguagem de interação.* O modelo ou linguagem de interação determina as atividades mentais e físicas que o usuário deve desempenhar, bem como os processos computacionais que o software da interface deve ter para interpretar os comandos e dados do sistema.

O modelo de interação pode ser caracterizado por um *padrão* e por um *estilo de interação.* O estilo diz respeito ao tipo de interação adotada que, normalmente, pode ser *linguagem de comando, menus, preenchimento de formulário, linguagem natural, e WIMP*. O padrão de um modelo de interação se refere a uma caracterização mais específica dos estilos através de regras que determinam atributos deste modelo. Nas

interfaces gráficas um padrão determina a aparência e o comportamento dos widgets. São exemplos de padrões de interfaces gráficas o *Motif* da ISO, o *OpenLook* da Sun Microsystems e o *Windows* da Microsoft.

O usuário precisa conhecer o modelo de interação para poder interagir como o sistema. Moran denomina-o de modelo conceitual do usuário e afirma que o design da interface é o design deste modelo [Moran 81]. Entretanto é importante diferenciar o modelo conceitual do usuário do modelo que é formalmente definido através da linguagem de programação que o implementa. Esta discussão é extremamente importante em nossa abordagem e voltará ser discutida no capítulo 4.

Modelo de Usabilidade

O conceito de *usabilidade* pode ser visto como a qualidade da utilização de sistemas computacionais a ser medida e analisada empírica e objetivamente [Butler 96]. Eason sugere que a usabilidade de um sistema depende não apenas da natureza do usuário, mas também das características da tarefa e do sistema. Isto significa que as variáveis das tarefas, do sistema e do usuário todas se combinam para determinar a usabilidade do sistema [Eason 84].

Adler e Winograd apresentam uma proposta mais radical e inovadora para usabilidade [Adler & Winograd 92]. Do ponto de vista deles, ela deve ser vista como a qualidade que ao mesmo tempo satisfaz necessidades do usuário, se acopla às suas capacidades e conhecimentos, considera o impacto da tecnologia no contexto de trabalho e integra o usuário em tal contexto. O desafio para usabilidade requer o design de equipamentos integrados ao ambiente de trabalho de maneira a aumentar as capacidades (principalmente as intelectuais) de usuários, considerando-os como pessoas inteligentes capacitadas para compreensão, aprendizado, interpretação e expressão, ao invés de desmerecê-los, considerando-os como “idiotas” destinados a trabalhos mecânicos. Neste sentido, sistemas computacionais devem ser vistos como artefatos cognitivos, ferramentas intelectuais ou mídia, no sentido introduzido no início deste trabalho.

Shackel argumenta que as características do ambiente e as situações de uso são também fatores determinantes [Shackel 89]. Shackel apresenta uma definição operacional de usabilidade, na qual requisitos devem ser especificados e avaliados posteriormente. Os

requisitos que devem ser atingidos são: *Facilidade de aprendizado, Eficácia, Flexibilidade e Atitude*, também citados em [Dix et al. 93]. Dentre estes fatores vamos considerar apenas a *facilidade de aprendizado*, que diz respeito ao conhecimento que o usuário deve adquirir para utilizar o sistema.

O conhecimento necessário para o usuário adquirir competência para utilizar o sistema, chamado por [Moran 81] e [Norman 86b] de modelo conceitual do usuário, envolve tudo aquilo o que se pode fazer - *a funcionalidade* - e como se pode interagir - o *modelo de interação*. Chamamo-o de *modelo de usabilidade* uma vez que ele é fundamental para a usabilidade do sistema. O design da aplicação de software é fundamentalmente o design do modelo de interação e do modelo funcional da aplicação, ou seja, do modelo de usabilidade.

O Apoio ao Design de Interfaces de Usuário

Nossa abordagem visa associar fundamentos teóricos com a prática de design. Esta integração essencial, no entanto, ainda não foi atingida satisfatoriamente. A indústria tem conduzido o processo de design baseado em diretrizes, princípios e ferramentas que são muito pouco baseados em teoria [Microsoft 95, Apple 92]. Nesta seção vamos mostrar as diversas formas de apoio que se pode dar ao processo de design. No capítulo seguinte vamos mostrar como teorias podem contribuir neste processo.

O apoio ao design pode ser dado através de modelos do processo e da instrumentação prática e teórica ao processo. Um modelo do processo ou método determina quais as atividades a serem realizadas, as etapas nas quais elas são desempenhadas, quem as desempenha. A instrumentação prática e teórica apóia as diversas atividades do design através de técnicas, ferramentas de software, formalismos e modelos teóricos dos fenômenos da interação.

Modelos do processo de design de interfaces

Identificamos na literatura duas categorias de ***modelos do processo***. Alguns autores argumentam que o design pode ser conduzido como um processo *iterativo-evolutivo baseado em prototipação* [Apple 92; Dix et al. 93; Hix & Hartson 93, Windows 95; Butler 96]. Neste processo, as atividades de design estão baseadas em princípios, regras e diretrizes, e o protótipo da interface deve ser avaliado experimentalmente. Outros

argumentam que as interfaces devem ser desenvolvidas seguindo uma *metodologia* sistemática no sentido de uma engenharia de interfaces, integrada com a engenharia de software [Curtis & Hefley 94; Lee 93].

Nos ***modelos iterativos baseados em prototipação***, em cada passo da iteração as soluções são concebidas e implementadas em um protótipo da interface que permita a realização de testes de usabilidade. De acordo com os resultados da avaliação uma nova análise pode ser realizada e o design do protótipo deverá ser refeito ou complementado.

Uma proposta ilustrativa de ***metodologias*** é a *Engenharia de Interfaces de Usuário* que explicitamente apresenta um ciclo-de-vida estendido do software de maneira a incorporar atividades voltadas para o design e implementação de interfaces [Curtis & Hefley 94]. Existem diversos outros trabalhos nesta linha que podem ser categorizados como *metodologias integradas de desenvolvimento*. A metodologia USE (*User System Engineering*) integra análise estruturada com o design de diálogos [Wasserman 91]. [Sutcliffe & McDermott 91] também integram um método de desenvolvimento que abrange análise de usuários e tarefas, especificação de interface e design de diálogos. A metodologia MOST estrutura a análise de tarefas e a integra com especificações e métodos mais formais da engenharia de software. [Lee 93] descreve, passo-a-passo, um método orientado-a-objetos para desenvolvimento de aplicações com interfaces gráficas. Ele considera como modelos de objetos, tradicionais em engenharia de software, podem ser mapeados em widgets de ferramentas de interfaces.

As abordagens metodológicas acima sistematizam as etapas de desenvolvimento integrando o design com as outras atividades do desenvolvimento e muitas vezes argumentam a necessidade de se aplicar teoria à pratica. Entretanto elas não visam abordar diretamente quais as teorias necessárias ao design para usabilidade. O que estas metodologias têm em comum é estarem todas associadas a métodos tradicionais estendendo-os para incorporarem fatores humanos, como análise de tarefas e modelos de diálogos. A ênfase em todos os casos é no design e na implementação do sistema computacional como um todo.

Instrumentação de apoio ao design

A instrumentação ao processo de design pode ser classificada em quatro categorias: (1) *princípios, regras, diretrizes e padrões de design*; (2) *ferramentas de software de apoio à prototipação da interface;* (3) *formalismos de apoio à especificação* e (4) *modelos téoricos dos componentes e processos da interação*.

Princípios de design são conceitos, normalmente vagos, que dizem respeito a certas perspectivas e considerações a serem seguidas pelos designers. São exemplos de princípios os conceitos de *consistência, usuário no controle, feedback, simplicidade*, dentre outros [Apple 92]. ***Diretrizes de design*** são, geralmente, recomendações mais específicas e detalhadas sobre decisões a serem tomadas, provenientes de casos anteriores de sucesso. As diretrizes muitas vezes são acrescidas de explicações e outros comentários e que são adotadas em concordância por designers. ***Regras de design*** são séries de especificações de design para uma aplicação em particular, expressas explicitamente de maneira a não causar interpretações ambíguas [Lindgaard 94]. A idéia de ***padrões de design***, proveniente da área de Arquitetura [Alexander 77], tem por objetivo sistematizar a experiência de designers através de descrições de soluções aplicadas a problemas em determinados contextos. Padrões descrevem, através de expressões mais formais, elementos e estruturas de composição, oferecendo uma alternativa mais formalizada a princípios e diretrizes de design. Padrões de design também foram adotados em engenharia de software para o design orientado-a-objetos [Gamma et al. 93]. A proposta de *linguagens de design* caracteriza o design como um processo comunicativo, no qual a funcionalidade dos elementos do produto projetado devem ser interpretada por quem os utiliza [Rheinfrank et al. 92; Rheinfrank & Evenson 96].

Ferramentas de software de apoio à prototipação são bastante utilizadas no design da interface e do software como um todo como forma de verificação antecipada da usabilidade do software. Um protótipo de interface permitir a construção dos diversos objetos de interfaces, o seu layout e outros aspectos estéticos e a sua configuração dinâmica. Existem inúmeras ferramentas de prototipação de interfaces gráficas. Algumas ferramentas permitem apenas a elaboração de layouts de tela, enquanto outras são verdadeiramente linguagens de programação, permitindo não apenas o layout da

interface, mas também a implementação de um sistema piloto (tais como Borland Delphi, Microsoft Visual Basic). Para uma classificação mais abrangente pode-se consultar [Myers 95]. A maioria das ferramentas de software não oferecem apoio à especificação do modelo de interação que está associado à interface. Como já citamos anteriormente, uma das razões para estas limitações é a falta de base teórica. Um dos objetivos de nosso trabalho é mostrar como ferramentas de software comuns podem ser apoiadas por modelos e formalismos de base teórica.

Formalismos de especificação têm por objetivo a descrição abstrata do modelo de interação e da funcionalidade sem entrar em detalhes a respeito de aspectos de implementação dos objetos de interfaces e das funções do sistema. Nosso trabalho propõe formalismos para a especificação do modelo de usabilidade. Na próxima seção discutiremos brevemente as diferenças entre especificação funcional e especificação do modelo de usabilidade. Grande parte dos formalismos de especificação atuais estão baseados em modelos teóricos cognitivos. Estes modelos e os formalismos associados são discutidos no próximo capítulo.

Modelos teóricos visam descrever os elementos e processos envolvidos na interação usuário-sistema a partir da aplicação de fundamentação teórica provenientes de disciplinas específicas. Vimos que IHC reúne diversas disciplinas que podem ser aplicadas a cada um dos focos de interesse do processo de interação. As abordagens centradas no sistemas aplicam ciência da computação na elaboração de modelos do sistema e de seus processos de funcionamento. As abordagens centradas-no-usuário aplicam conhecimento das ciências humanas na descrição dos processos do usuário. Os modelos teóricos proporcionados por estas abordagens são a base para a elaboração dos formalismos de especificação. O próximo capítulo apresenta as principais abordagens teóricas discutindo os modelos e formalismos associados a cada uma delas.

Especificação do modelo de usabilidade

As decisões e soluções obtidas na atividade de design devem ser descritas em uma especificação. Na engenharia de software a *especificação de software* refere-se à descrição da funcionalidade, também chamada de especificação funcional. A

especificação funcional possibilita a descrição abstrata daquilo que o software deve fazer, independente de como será implementado.

A especificação, embora possa ser descrita em linguagem natural, deve ser realizada com o apoio de *linguagens de especificação* que podem ser formais ou semi-formais [Ghezzi 91]. São exemplos de linguagens de especificação funcional semi-formais os *diagramas de fluxo de dados, os diagramas entidade-relacionamento e os diagramas de transição de estados.* As linguagens de especificação formais garantem a precisão da descrição através da fundamentação matemática e de mecanismos de prova formal. Elas podem ser axiomáticas, algébricas ou baseadas em modelos. *VDM* e *Z* são exemplos bastante conhecidos de linguagens formais baseadas em modelos.

Na engenharia de software a especificação funcional está voltada para a descrição abstrata de todos os componentes (funções e estruturas de dados) a serem implementados. À medida que o tamanho e a complexidade do sistema cresce, o design e a especificação da estrutura global passa a ser mais importante do que a escolha dos algoritmos e estruturas de dados. A *especificação da arquitetura do software* tem por objetivo a descrição da estrutura global dos componentes [Shaw & Garlan 96]. A disciplina de *Arquitetura de software* tem por objetivo apresentar estilos, padrões, frameworks e modelos de arquitetura com o objetivo de apoiar a concepção da estrutura do software. A concepção e especificação da arquitetura introduz um nível de design intermediário entre a especificação funcional e a especificação de estruturas de dados e algoritmos.

No design de interfaces de usuário, além da especificação da funcionalidade, é preciso conceber e especificar o modelo de interação [Hix & Hartson 93]. A especificação da interação deve descrever as diversas maneiras que o usuário vai interagir com o sistema. São exemplos de linguagens de especificação da interação, *gramáticas, os diagramas de transição de estados, State-charts, UAN (User-Action Notation)* [Shneiderman 98]. Formalismos como *CSP (Communicating Sequential Processes)* e *Redes de Petri* também podem ser utilizados para a especificação da interação [Dix et al. 93]. No capítulo 3 apresentaremos outros formalismos de especificação desenvolvidos a partir de abordagens teóricas para o design de interfaces.

Da mesma forma que na especificação funcional, a especificação da interação deve ser independente da implementação da interface. Porém, é importante diferenciar a especificação do modelo de interação da especificação do software que vai implementar a interface [Hix & Hartson 93].

Uma das grandes vantagens da especificação é permitir que a funcionalidade e o modelo de interação possam ser avaliados por desenvolvedores e usuários através da construção de modelos e protótipos que permitem simulações e testes, antes mesmo de sua implementação.

Temos argumentado desde o início deste trabalho que a usabilidade depende tando da interatividade quanto da funcionalidade. Por este motivo, a especificação do modelo de interação e da funcionalidade devem ser bastante relacionadas entre si. Entretanto, tal relacionamento não é demostrado na maioria das abordagens de design. Pelo contrário, engenheiros de software têm ressaltado a necessidade da independência entre a interface e o núcleo funcional. Embora isto seja interessante no nível da arquitetura de software, isto é, os componentes do software que implementam o modelo de interação devem ser independentes dos componentes do núcleo funcional, no processo de interação eles são indissociáveis, como veremos no capítulo 5. Além disso, as abordagens para especificação funcional têm por objetivo a descrição abstrata dos componentes funcionais sem compromisso com a usabilidade.

Desta forma a perspectiva de design de software visando usabilidade apresentada em [Winograd 96] requer a ***especificação do modelo de usabilidade***, isto é, a especificação do modelo de interação e da funcionalidade aplicados ao domínio como uma solução-em-potencial. As ferramentas de apoio ao design de interfaces de usuário que apresentamos neste trabalho necessita de formalismos de especificação do modelo de usabilidade.

Avaliação Experimental da Usabilidade

Visando suprir a falta de uma abordagem baseada em modelos teóricos, alguns pesquisadores argumentam a necessidade de maior rigor no processo de avaliação com o objetivo de garantir a usabilidade de um sistema.

A avaliação experimental corresponde a especificar requisitos de usabilidade e avaliá-los empiricamente num processo iterativo de design e desenvolvimento [Shackel 89]. Segundo Lindgaard, devido à imprevisibilidade do comportamento e desempenho do usuário em diferentes situações e condições emocionais, a melhor alternativa para se assegurar a usabilidade de sistemas computacionais é testá-los com uma amostragem de pessoas que sejam representativas dos possíveis usuários reais [Lindgaard 94]. A idéia de se especificar as métricas de usabilidade tem sido bem aceita nos meios acadêmicos e industriais. O principal problema está em como determinar quais os requisitos de usabilidade que um determinado sistema deve satisfazer e como isto está diretamente ligado à interface de usuário. A própria definição de usabilidade é considerada vaga.

O processo de avaliação experimental é baseado em técnicas de psicologia experimental, matemática e estatística. Técnicas de elicitação[6] de requisitos da análise de sistemas são também aplicadas [Lindgaard 94]. A psicologia fornece uma importantíssima contribuição na aplicação de métodos de avaliação empírica e análise de dados com respeito a princípios e alternativas de design [Landauer 88].

Alguns autores utilizam o termo *engenharia de usabilidade* para métodos com enfoque na avaliação experimental [Nielsen 93, Butler 96]. A engenharia de usabilidade é o processo no qual a usabilidade de um produto é especificada quantitativamente no início do processo de desenvolvimento.

Um problema com esta ênfase está na aplicação dos resultados dos testes em futuros projetos. Eles indicam se o sistema projetado satisfaz ou não aos requisitos previamente definidos, entretanto não indicam em quais aspectos o sistema precisa ser reprojetado. Em outras palavras, as abordagens experimentais sugerem um método prático para se medir informações quantitativas a respeito da interação, mas não indicam como as informações qualitativas podem ser utilizadas para se modificar e melhorar o sistema [Booth 88]. As informações quantitativas não revelam todas as relações necessárias para se obter objetivamente explicações e previsões acerca de causas e efeitos relativos a decisões de design. Não basta identificar que um design é ruim porque um

[6]Apesar de o termo dicionarizado em português ser *eliciação/eliciar*, estaremos usando a corruptela *elicitação/elicitar* por seu caráter técnico específico e seu status real de jargão na área de Engenharia de Software.

teste revelou, por exemplo, que em 90% dos casos ocorrem erros do usuário. É preciso entender as relações de um aspecto revelado por uma avaliação empírica com o design da interface.

Este trabalho não aborda aspectos da avaliação empírica da usabilidade. Embora o nosso objetivo esteja na fundamentação teórica de apoio ao design, é importante que os sistemas desenvolvidos com o apoio teórico que apresentamos seja avaliado através de testes de usabilidade sistemáticos.

Resumo

Nosso objetivo com este capítulo foi identificar os focos de interesse do nosso trabalho, discutir alguns conceitos e ressaltar as diferentes formas de apoio ao design de interfaces, enfatizando a importância das ferramentas e dos formalismos de especificação baseados em modelos teóricos.

Inicialmente apresentamos a área onde inserimos o problema que é tratado nesta tese. A IHC foi apresentada como um campo multi-disciplinar que se caracteriza pela aplicação de teorias de diversas outras disciplinas na explicação e previsão de fenômenos da interação entre seres humanos e sistemas computacionais e no design de tais sistemas. A semiótica é mais uma área do conhecimento cujos conceitos podem ser aplicados em IHC.

Além da diversidade de disciplinas, pudemos identificar, ainda, diferentes enfoques para o estudo do diversos fenômenos associados. Nosso enfoque está no domínio conceitual, ou comportamental, do processo de design e interação e não em aspectos de sua implementação. Na seção 2.2 apresentamos os componentes conceituais de um sistema interativo que são abordados no design para usabilidade. Como o objetivo desta tese é aplicar conceitos téoricos ao processo de design de interfaces, integrando teoria e prática, discutimos ainda, na seção 0, como os processos de design e desenvolvimento têm sido conduzidos e quais as ferramentas e formalismos que apóiam tais processos. Um problema identificado com os formalismos de especificação é a falta de integração entre o modelo de interação e de funcionalidade visando usabilidade.

ABORDAGENS TEÓRICAS PARA O DESIGN DE INTERFACES DE USUÁRIO

Vimos no capítulo anterior que a IHC, por ser um campo de estudos recente e que agrupa pesquisadores de diversas áreas, ainda não apresenta teorias consagradas para o design, desenvolvimento e avaliação de interfaces de usuário. O que podemos identificar são diferentes propostas elaboradas por pesquisadores que aplicam conhecimentos provenientes de disciplinas específicas nas diversas etapas do design. Chamaremos estas propostas de *abordagens teóricas para o design de interfaces de usuário*. A Engenharia Semiótica apresenta uma abordagem que aplica conceitos de semiótica que permitem identificar ou reinterpretar fenômenos, enunciar problemas e descobrir soluções.

Para melhor se avaliar nosso trabalho é preciso compará-lo com outras abordagens. Ao mesmo tempo que é laborioso um levantamento exaustivo de propostas teóricas, é também difícil uma categorização homogênea a partir de critérios comuns. Neste capítulo vamos analisar estas abordagens teóricas através de critérios que revelem semelhanças e diferenças com os pontos-chave enfatizados na nossa abordagem.

Uma Classificação para as Abordagens Teóricas

Na seção seguinte, mostraremos como alguns autores apresentam e classificam abordagens téoricas para IHC. Em seguida, mostraremos os critérios que utilizamos para categorizá-las.

A Classificação de Eberts

Eberts apresenta quatro abordagens para o design de interfaces: *a experimental, a cognitiva, a de modelos de previsão e a antropomórfica* [Eberts 94]. A abordagem experimental considera o processo de desenvolvimento como uma atividade de

experimentação seguida de avaliação empírica. Esta abordagem será discutida mais adiante. A abordagem cognitiva, como é a de [Norman 86], se preocupa com o processo de aquisição, pelo usuário, do modelo mental associado ao modelo conceitual do sistema. A abordagem de modelos de previsão é aquela cujo desenvolvimento está baseado em modelos cognitivos de usuários genéricos que permitem ao designer prever o desempenho de usuários durante a interação. A abordagem antropomórfica visa a elaboração de sistemas computacionais com características as mais próximas possíveis das humanas, de maneira a facilitar o processo de interação com os seus usuários.

As categorias propostas por Eberts não seguem um critério de categorização homogêneo. As abordagens de modelos de previsão e cognitivas se diferenciam não pela perspectiva teórica adotada, ambas cognitivas, mas, no caso da primeira, pela utilização de modelos cognitivos genéricos dos usuários como base para a tomada de decisões no processo de design. A abordagem cognitiva de Eberts considera que o design deve ser conduzido dentro da perspectiva de como construir o modelo cognitivo na mente do usuário. A abordagem experimental também não se distingue das outras por considerar uma proposta teórica alternativa. Ela se diferencia justamente por não utilizar uma proposta teórica como base para o design, enfatizando na avaliação experimental.

O autor reconhece que as divisões entre as quatro abordagens não é sempre clara e alguns trabalhos podem cair em mais de uma categoria. Seu objetivo é organizar as diferentes perspectivas de estudos teóricos quanto a como podem ser aplicados no design. O papel do designer muda em cada uma delas. Desta forma, o autor argumenta que as quatro abordagens devem ser integradas, cada uma enfocando uma etapa do processo de design.

Critérios para Análise das Abordagens

Os critérios que escolhemos visam permitir uma avaliação crítica da nossa abordagem de Engenharia Semiótica. O primeiro deles considera qual a *disciplina* ou *área de pesquisa* que oferece os fundamentos teóricos aplicados em IHC. Este é o principal critério, pois disciplinas são caracterizadas por correntes teóricas (escolas) que determinam como problemas e soluções podem ser identificados. O segundo critério visa identificar quais modelos teóricos são propostos para os fenômenos envolvidos (design

da interface e interação usuário-sistema). O terceiro critério avalia como resultados teóricos são aplicados à prática de design, isto é, se na forma de princípios e diretrizes, de ferramentas de software, de formalismos de especificação ou de modelos teóricos (ver capítulo 2). Não estamos considerando trabalhos cujas soluções são aplicadas a outras etapas do desenvolvimento que não a de design, tais como análise de requisitos e avaliação da usabilidade. Enfatizamos principalmente os trabalhos que abordam que visam os aspectos de *facilidade de aprendizado* dentre os critérios de usabilidade.

Em resumo, estes critérios são:

- a disciplina teórica que fundamenta os conceitos;
- os principais modelos propostos para os fenômenos envolvidos;
- a contribuição para a prática de design de interfaces de usuário.

Principais abordagens identificadas

Identificamos inúmeros trabalhos que visam aplicar disciplinas teóricas ao estudo da interação entre usuários e sistemas e ao design de interface. A maioria deles se enquadram numa abordagem geral que chamamos de *cognitivas*, por aplicarem disciplinas da chamada ciência cognitiva. Um outro grupo se destaca por estar utilizando a mesma disciplina teórica que motiva o nosso trabalho. Um grande destaque será dado a este segundo grupo com o objetivo de salientar os principais aspectos que o nosso trabalho visa complementar.

Além destas duas grandes classes, dois trabalhos individualmente merecem destaque devido a sua grande repercussão na literatura especializada. Tal como a nossa abordagem, eles apresentam novas perspectivas e modelos teóricos a partir da aplicação de conceitos teóricos provenientes de disciplinas não tradicionalmente associadas a IHC: um é o de Bodker [Bodker 89] e o outro é o de Laurel [Laurel 91].

Bodker aplica a *teoria da atividade humana* de Leontjew em IHC e desenvolve um *framework* para o design de interfaces de usuários [Bodker 89]. Este *framework* aborda o papel da interface de usuário no trabalho humano. Seu ponto central é o de que a interface não pode ser vista independentemente da atividade de uso.

Já Laurel apresenta uma perspectiva na qual o processo de interação é visto como uma *encenação* onde o usuário e os processos computacionais são agentes que manipulam objetos e se comunicam entre si [Laurel 91]. A partir desta perspectiva ela desenvolve uma abordagem baseada na teoria de *dramaturgia* de Aristóteles, na qual a interface é o palco ou ambiente para as interações. Desta forma, o designer da interface é o autor de uma história interativa, com cada elemento desempenhando o seu papel, e que considera as diversas situações de uso, num contexto dinâmico.

A abordagem de Laurel apresenta semelhanças com a Engenharia Semiótica, que será vista adiante, na qual o que chamaremos de *Mensagem do Designer* é um signo com esta característica interativa e dinâmica.

As Abordagens Cognitivas

As abordagens dominantes que têm caracterizado IHC são as cognitivas [Preece et al. 94]. Os resultados delas são de longe mais numerosos do que os de qualquer outra. Ao mesmo tempo que é bastante difícil uma comparação com todos os trabalhos já produzidos nestas abordagens, um avaliação generalizada pode ser superficial. Analisamos os principais aspectos na maioria destes trabalhos visando identificar suas principais contribuições e os limites que tornam necessária a abordagem apresentada em nosso trabalho.

As disciplinas teóricas das abordagens cognitivas

A abordagens cognitivas têm por objetivo a aplicação de estudos de psicologia cognitiva, ciência cognitiva e inteligência artificial para a compreensão das capacidades e limitações da mente dos usuários.

A ciência cognitiva tem sido uma importante fonte de conhecimento teórico para IHC. Trata-se de um campo multi-disciplinar para a compreensão de processos cognitivos que abrange disciplinas com filosofia, psicologia, antropologia, lingüística, inteligência artificial e neurofisiologia. Ela surgiu a partir dos trabalhos em lingüística de Chomsky [Chomsky 53] e do trabalho de Simon & Newell em inteligência artificial [Gardner 85]. Seu objetivo é oferecer uma abordagem mais subjetiva (interpretativa) dos processo mentais, examinando o papel do conhecimento, compreensão, aprendizado e significado

na interação usuário-sistema. O espectro de aplicações da ciência cognitiva em IHC ainda está em discussão [Pollitzer & Edmonds 96].

As abordagens cognitivas visam oferecer modelos teóricos para a compreensão do que ocorre na interface de usuário, a partir de uma perspectiva centrada nos aspectos cognitivos do usuário [Booth 88, Preece et al. 94]. Em geral, cognição refere-se ao processo pelo qual se pode adquirir conhecimento. O foco está em modelos que visam descrever os processos mentais para o aprendizado - a aquisição dos modelos conceituais do sistema - e para o desempenho da interação - o mapeamento dos modelos de tarefas com modelos do sistema. A aplicação destes conhecimentos se justifica se tomarmos como base que o desempenho dos usuários durante a interação tem como origem os processos mentais (conhecimento) a respeito das tarefas e do sistema [Norman 86].

A formulação do problema

A perspectiva cognitiva tenta caracterizar os modelos e as atividades mentais relacionados com a interação usuário-sistema e assim poder definir melhor os problemas e apresentar suas soluções. Vimos na introdução deste trabalho que o aspecto de usabilidade que estamos tratando envolve principalmente a facilidade de aprendizado dos modelos de interação e de funcionalidade.

Inicialmente apresentaremos dois trabalhos clássicos que formulam este problema a partir de uma perspectiva cognitiva. Mais adiante discutiremos como alguns trabalhos apresentam soluções.

Norman considera que as pessoas interagem com o sistema a partir da formulação de um modelo mental do sistema. Este modelo permite ao usuário mapear suas metas em comandos e funções do sistema (ver figura 0-1). Segundo esta proposta, o papel do designer está em desenvolver uma interface que permita ao usuário, durante o processo de interação, adquirir um modelo mental correspondente ao modelo do sistema [Norman 86]. Para isto são necessários modelos cognitivos das atividades mentais. Norman considera que este modelo mental do usuário é adquirido durante a interação com a *imagem do sistema* - a interface, o sistema de ajuda e toda a documentação do sistema.

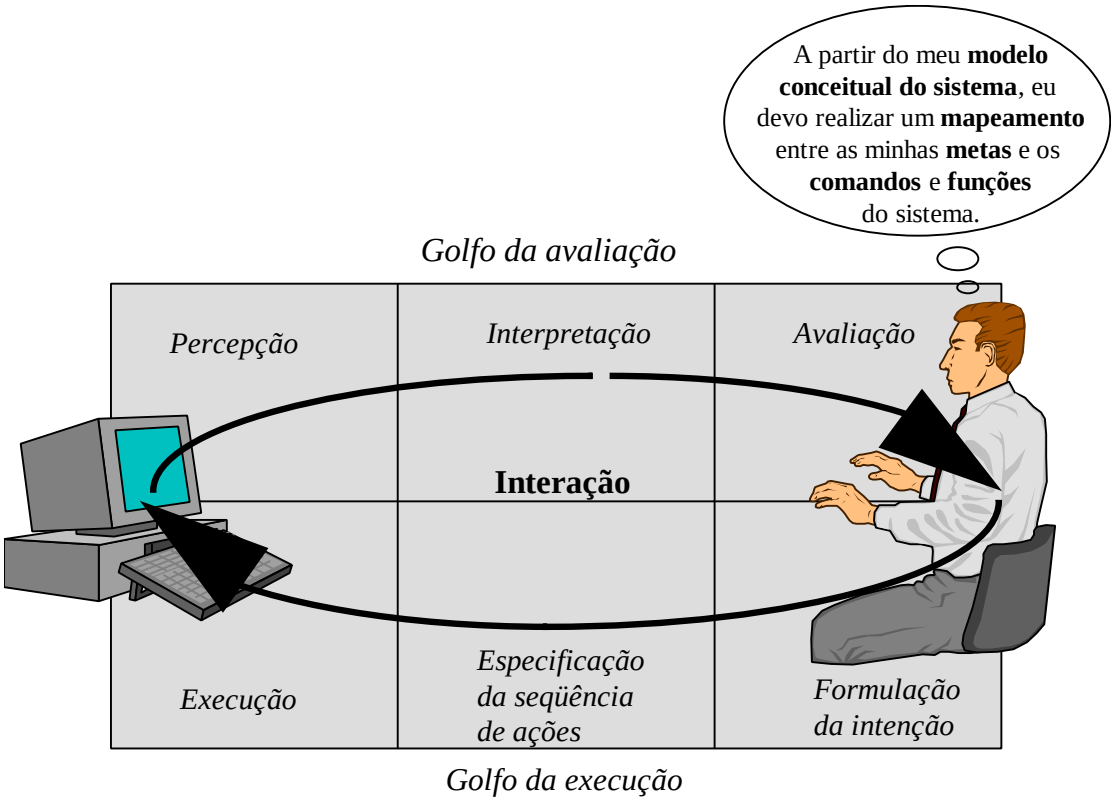


Figura 0-1: A perspectiva da abordagem cognitiva de [Norman 86].

A proposta de Norman argumenta que IHC necessita de *teorias da ação humana* e é sugerido que estas atividades mentais podem ser estudadas por modelos cognitivos. O esquema da *teoria da ação* propõe que a interação usuário-sistema é desempenhada num ciclo-de-ação com sete etapas e dois "golfos" a serem atravessados. A travessia dos golfos é iniciada a partir do *estabelecimento da meta* que é uma atividade cognitiva motivada pelo contexto do usuário. Um deles é o *golfo da execução* e envolve as etapas de *formulação da intenção, especificação da seqüência de ações* e *atividade física (execução).* O outro é o *golfo da avaliação* e deve ser atravessado pelas etapas de *percepção, interpretação e avaliação da meta* (ver Figura 0-1).

A teoria da ação de Norman trata das atividades mentais que governam o processo de interação. O próprio autor considera que esta teoria é uma aproximação. Sua principal vantagem está em aumentar a compreensão teórica do processo e possibilitar o surgimento de novas teorias.

O modelo de Norman oferece a perspectiva teórica a partir da qual os problemas da interação são formulados. Na análise da literatura identificamos dois subproblemas: o *mapeamento tarefa-ação* que o usuário precisa desempenhar (a travessia do golfo da execução) [Payne & Green 86]; e o problema da *avaliação* (a travessia do golfo da avaliação). A maioria dos trabalhos estão direcionados na solução do mapeamento tarefa-ação. As abordagens cognitivas para interfaces gráficas ressaltam a importância da *avaliação* na realização do mapeamento.

O mapeamento tarefa-ação é a atividade mental que o usuário precisa desempenhar para associar conceitos mentais a respeito de tarefas a serem realizadas (o modelo conceitual da tarefa) com os conceitos mentais sobre a seqüência de ações na interface (o modelo conceitual de interação). Esta atividade requer conhecimento sobre o modelo de interação e a funcionalidade do sistema.

Para as abordagens cognitivas os problemas associados às atividades mentais requerem o estudo do processo de *recordação* de comando e regras e a *interpretação* do signos utilizados na interação.

O mapeamento tarefa-ação pode ser analisado através das *distâncias semânticas e articulatórias* [Hutchins et al. 86]. A distância semântica avalia a separação entre as metas/tarefas do usuário e a funcionalidade do sistema a elas associada, isto é, se existe um comando no modelo de interação cujo significado (resultado ou efeito) seja aquele pretendido pelo usuário. Uma distância pequena significa que existe um comando (quase que) diretamente associado à meta, enquanto que uma distância grande indica que o usuário precisa quebrar metas em submetas e realizar um planejamento de tarefas. A distância articulatória avalia o relacionamento entre o significado (resultado ou efeito) percebido de um comando e a forma da seqüência de ações (o comando) tal como se disponibiliza para o usuário.

Visando exemplificar este aspecto, vamos considerar um usuário de um editor de textos qualquer que tem como meta *melhorar o formato do texto.* Esta meta é bastante abrangente, pois envolve diversas operações do sistema. Vamos supor que o editor de texto dispõe de modelos de formatação prévios e que o usuário pode comandar a operação de *formatar o texto seguindo o modelo X* pressionando apenas duas teclas.

Neste caso, o sistema possui uma operação que satisfaz diretamente a meta do usuário (distância semântica mínima) e para a qual existe um comando com apenas duas ações (pequena distância articulatória). Caso o sistema possua apenas as funções de formatação básicas (formatar caracteres e parágrafos), o usuário precisa realizar um esforço de planejamento especificando submetas mais específicas para os quais existam funções no sistema. Neste caso existe uma considerável distância semântica a ser percorrida. Especificando as submetas em termos de *formatar fonte negrito* ou *centralizar parágrafo,* o usuário pode encontrar comandos que podem ser realizados por uma única ação (presionar um botão, por exemplo), o que caracteriza uma distância articulatória mínima. Para submetas do tipo *formatar distância do parágrafo em relação as bordas* pode existir uma operação no editor que a realize, mas cujo comando requer a especificação de diversos parâmetros como as margens esquerda, direita, superior e inferior. A distância semântica é, neste caso, mínima, enquanto que a articulatória requer um esforço grande a ser percorrido uma vez que diversas ações são necessárias para se compor o comando.

O problema da avaliação também deve ser estudado através das distâncias semântica e articulatória. A distância articulatória permite avaliar o processo de interpretação, i.e., o significado atribuído pelo usuário ao que lhe está sendo mostrado. A distância semântica tem por objetivo medir se o que foi interpretado está direta ou indiretamente relacionado com a meta pretendida. Ela é medida através de modelos cognitivos do processo de interpretação.

A perspectiva cognitiva mostra que a *facilidade de aprendizado* deve ser solucionada com o design de interfaces que proporcionem o menor esforço cognitivo para as atividades no mapeamento tarefa-ação e na aquisição do modelo conceitual. O objetivo destas abordagens é descobrir o conhecimento teórico a respeito das atividades mentais que orientam o design da interface para usabilidade. Veremos que nestas abordagens existem diferentes estratégias de solução, atendendo a sub-objetivos distintos deste quadro geral.

A formulação do problema na abordagem cognitiva que acabamos de apresentar é extremamente importante para a compreensão deste trabalho. Nossa abordagem apresenta

propostas complementares para algumas limitações das soluções da abordagem cognitiva. Na próxima seção apresentamos algumas destas soluções e discutimos as suas limitações.

Estratégias de Solução

A estratégia de solução das abordagens cognitivas para os problemas mencionados acima consiste na elaboração de modelos cognitivos genéricos que permitam aos designers entender os processos cognitivos humanos usados na interação e realizar experimentos ou previsões com estes modelos (ver figura 0-2). A idéia básica é que modelos cognitivos que descrevem os processos e estruturas mentais, tais como recordação, interpretação, planejamento e aprendizado, podem indicar para pesquisadores e projetistas quais as propriedades que os modelos de interação devem ter de maneira que a interação possa ser desempenhada mais facilmente.

A psicologia cognitiva contribui com esta abordagem na aplicação de idéias da psicologia da "*máquina mental teórica*" para a construção de novos modelos e ferramentas de análise e engenharia [Landauer 88]. Estes modelos são construídos a partir de resultados experimentais da avaliação do comportamento humano. Um dos trabalhos pioneiros na aplicação de psicologia cognitiva a IHC é o de [Card, Moran & Newell 83], cujo objetivo é a utilização de técnicas cognitivas para a construção modelos que possibilitam aos designers fazer previsões de desempenho dos usuários e, conseqüentemente, oferecer condições para a tomada de decisões.

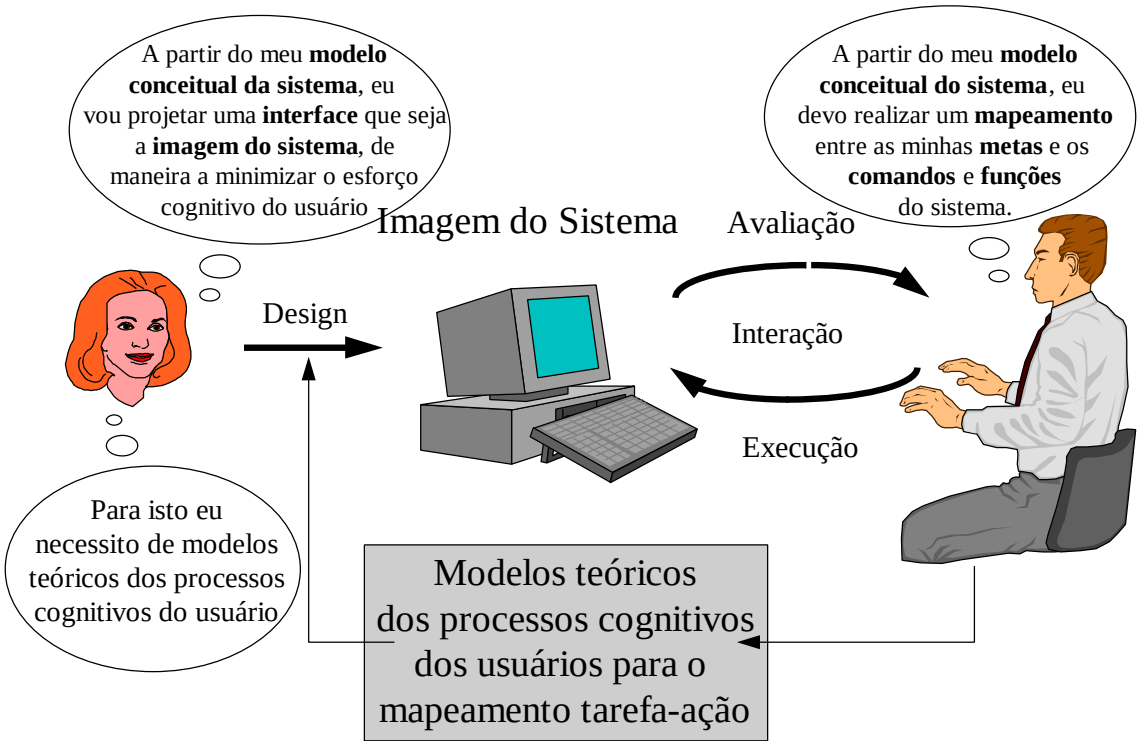


Figura 0-2: A estratégia de solução das abordagens cognitivas

Vamos a seguir analisar as principais estratégias de soluções das abordagens cognitivas, classificando-as em *empíricas, baseadas em modelo de desempenho, baseadas em modelo de competência,* e *baseadas no display.*

As Abordagens Cognitivas Empíricas

A abordagens cognitivas baseadas na análise empírica visam o estudo de atividades cognitivas na interação usuário-sistema através da aplicação de métodos e técnicas da psicologia experimental na análise do comportamento do usuário no processo de interação. O objetivo é identificar através de uma avaliação empírica características, conceitos e modelos cognitivos dos usuários que devam ser incorporados aos modelos de interação das interfaces de usuários.

Por exemplo, experimentos em psicologia indicam que os problemas do mapeamento e da aquisição do modelo conceitual podem ser resolvidos através de modelos de interação com *consistência,* com *interatividade* e *vocabulário significativo* [Howes & Young 96]. Um mapeamento tarefa-ação é consistente se propriedades estruturais do modelo de interação como a sintaxe são compartilhadas. A interatividade

refere-se ao papel que os dispositivos de interface e as informações no display exercem na execução do mapeamento correto. O problema do vocabulário consiste em elaborar representações significativas para os comandos da interface.

Outros autores apresentam estas propriedades de forma diferente. O vocabulário é visto como um problema de escolha de *mnemônicos*, enquanto que a consistência semântica é descrita como um problema de *regularidade* e o problema da consistência é visto como a manutenção do mesmo mapeamento tarefa-ação ao longo de diversos contextos de aplicação [Lee et al. 94]. A *ortogonalidade* é um princípio aplicável a propriedades de linguagens comprovada empiricamente que dá às linguagens de interação uma estrutura mais simétrica, possibilitando um menor esforço cognitivo no mapeamento tarefa-ação.

A análise experimental também pode ser utilizada como forma de descobrir e validar os modelos cognitivos. Aliás a própria ciência cognitiva ganhou impulso com o argumento de que se deveria elaborar modelos cognitivos a partir dos resultados empíricos da psicologia [Simon & Newell 72].

As Abordagens Cognitivas Baseadas em Modelos

Tais abordagens têm por objetivo a elaboração dos modelos cognitivos[7] necessários à explicação e previsão dos processo mentais na interação. Os modelos cognitivos podem ser aplicados de várias maneiras em IHC. Podemos identificar duas linhas básicas, com base em duas teorias pioneiras em ciência cognitiva. Uma delas, a mais comum, consiste em aplicar modelos cognitivos de forma a fazer previsões sobre o comportamento humano em todos os níveis de interação. Estes modelos são considerados *modelos cognitivos de desempenho* e estão baseados nos trabalhos de *Simon & Newell* sobre o processamento humano de informações para a resolução de problemas [Simon & Newell 72]. Eles apresentam uma arquitetura cognitiva da capacidade humana de processamento de informações para resolução de problemas. O processamento de informações para a resolução de problemas ocorre através da aplicação *regras de*

[7]Como a IHC e a ciência cognitiva são áreas multi-disciplinares, existem diversos significados que podem ser atribuídos a modelos cognitivos, mentais, conceituais e semânticos. Neste trabalho, o termo *modelo cognitivo* se refere aos tipos de representações simbólicas que se pode fazer a respeito dos processos cognitivos humanos. Trata-se de modelos simbólicos *sobre* a representação mental (modelo mental), o que

produção que realizam a busca de soluções no *espaço do problema*. O principal e mais representativo trabalho nesta linha visa aplicar psicologia à construção de modelos dos usuários em diversos níveis do processo de interação [Card et al. 83].

A segunda linha derivou-se do trabalho de *Chomsky* em lingüística e tem por objetivo elaborar *modelos da competência* do usuário para a interação com o sistema [Chomsky 65]. Estes modelos visam descrever conhecimento ou arquiteturas de conhecimento que os usuário devem ter para desempenhar as atividades de interação.

Modelos Cognitivos de Desempenho

Esta linha enfoca a aplicação de teorias da psicologia na elaboração de modelos cognitivos com ênfase na análise e previsão do desempenho humano, tanto no mapeamento tarefa-ação quanto na avaliação da interação. O principal objetivo é aplicar modelos de desempenho ao design de interfaces através de previsões do desempenho cognitivo e motor dos usuários. Estes modelos possibilitam ao designer experimentar diversas alternativas de design alterando valores de suas variáveis. Desta forma é possível prever o comportamento de usuários com diversas alternativas de interfaces. O designer ganha conhecimento a respeito da carga cognitiva e física imposta aos usuários para a realização do mapeamento tarefa-ação.

Com base na proposta de *Simon e Newell* [Simon & Newell 72] foi desenvolvido um dos trabalhos pioneiros da abordagem cognitiva para IHC: *O Modelo do Processador Humano (MHP)* [Card Moran & Newell 83]. O modelo *GOMS*[8] está baseado nesta teoria e se propõe a ser um formalismo para modelar a estrutura de metas, métodos e operações que o usuário precisa executar para realizar as tarefas. Outro modelo relacionado com o MPH é o *KLM*[9] *(Keystroke-level Model*) cujo objetivo é descrever atividades físicas dos usuários na interação (modelo físico).

A *Teoria da Complexidade Cognitiva (CCT)* estende a idéia de decomposição de metas visando dar um maior poder de previsão ao modelo [Kieras & Polson]. A descrição de metas é semelhante à do GOMS, mas é expressa através de *regras de produção*. A

---

deve ser diferenciado de modelos mentais, conceituais e semânticos, usados geralmente para tratar diretamente a representação.

[8]GOMS - do inglês *Goals, Operat~~ion~~ors, Methods and Selection ~~r~~Rules* (Metas, Operações, Métodos e Regras de Seleção).

PUM (Modelos de Usuários Programáveis) é uma arquitetura que modela usuário e habilita designers a simularem o desempenho de usuários em várias tarefas ([Young et al.], citado em [Dix et al. 93]).

Embora as pesquisas baseadas em psicologia cognitiva, a partir do marcante trabalho de [Card, Moran & Newell 83], tenham dominado a cena durante boa parte da década de 80, no início da década de 90, [Carroll *et al.* 91] consideraram que esta disciplina não apresentava respostas imediatas para muitos aspectos e problemas de IHC. Especialistas em psicologia sabem que o comportamento humano é imprevisível, não confiável e bastante variável de acordo com as circunstâncias [Lindgaard 93]. O desempenho também varia de acordo com o estado emocional de cada um. A maioria dos modelos cognitivos desenvolvidos não considera a influência de circunstâncias e aspectos emocionais no desempenho de usuários. Modelos de tarefas que presumem que o usuário os execute fielmente, restringem o que o usuário pode fazer e criar. Suchman questiona a aplicação destes modelos com a argumentação de que pessoas não agem baseadas em planos pré-determinados [Suchman 87]. Segundo a autora, os planos são refeitos dinamicamente de acordo com as circunstâncias contextuais nas quais estão envolvidos os usuários. Booth considera ainda o problema do grão de análise. Isto significa que, além da imprecisão inerente aos modelos em fazer previsões de desempenho, o grão de análise empregado, é muitas vezes determinante nesta precisão [Booth 88].

Modelos Cognitivos de Competência

A teoria lingüística de Chomsky diz que a *competência lingüística* de uma pessoa - as estruturas cognitivas inatas e o conhecimento adquirido a respeito da linguagem - determina e é determinada pelo desempenho *lingüístico* - a experiência interativa. Segundo Chomsky, o modelo básico da capacidade lingüística humana possui três componentes principais: o *modelo de desempenho,* o *modelo de competência* e *fatores extra-lingüísticos.* No centro deste modelo está o componente da competência lingüística que um usuário da linguagem deve ter. Ele é, junto com os fatores extra-lingüísticos, a base na qual o mecanismo de desempenho possibilita as atividades lingüísticas de compreensão e produção de sentenças. A competência é responsável pela intuição sobre a semântica, sintaxe e léxico da linguagem.

[9]KLM - do inglês *Keystroke Level Model* (Modelo no Nível de Digitação)

Aplicando a teoria do Chomsky à interação usuário-sistema, alguns pesquisadores adotam os termos modelo de competência e de desempenho em IHC [Dix et al. 93; Green et al. ; Poitrenaud 96.] para se referirem, respectivamente, a modelos de conhecimento a respeito da interface e a modelos do desempenho na realização de tarefas na interface. Consideraremos que este termo é empregado de maneira metafórica apenas, sem maiores compromissos com a teoria do Chomsky. Os modelos citados na seção anterior são modelos do desempenho. Existem porém alguns modelos que são tanto de competência como de desempenho, como o PROCOPE [Poitrenaud 96] e o trabalho de [Kitajima & Polson 95].

Os trabalhos nesta linha visam oferecer ao designer modelos ou arquiteturas cognitivas (genéricas) da competência dos usuários de maneira que ele disponha de conhecimento sobre porque certos modelos de interação permitem mapeamentos tarefas-ações que são mais facilmente compreendidos e realizados do que outros.

Os principais modelos e formalismos para a competência estão na forma de *gramáticas de ação* [Dix et al. 93, Poitrenaud 96]. As gramáticas de ação são modelos gramaticais para o mapeamento entre tarefas dos usuários. Os autores argumentam que estas gramáticas são representações mentais do modelo de interação [Green & Payne 86, Howes & Young 96]. Técnicas de representação como *BNF (Backus Naur Form)*, *Set Grammars* e *Task-Action Grammar (TAG)* [Payne & Green 86]) são aplicações dos conceitos de Chomsky em IHC, caracterizando modelos da representação mental das linguagens de interfaces (ou de outras linguagens artificiais).

Um dos formalismos mais conhecidos para especificação baseada em gramáticas de ação é a *Task-Action Grammar* (TAG). A TAG é um formalismo que permite elaboração de linguagens de comandos a partir de modelos de tarefas [Payne & Green 86]. A TAG captura consistência do modelo de interação pela representação da competência de um usuário especialista através de uma notação formal. O objetivo a ser alcançado é facilitar o aprendizado e melhorar o desempenho do usuário. Assim, a ênfase está em resolver os problemas relativos à consistência. A TAG funciona como uma meta-linguagem que constitui um modelo de representação mental de linguagens de tarefas e é,

portanto, um modelo de competência. A idéia do mapeamento tarefa-ação é semelhante ao mapeamento entre os níveis conceituais e os de comunicação.

Segundo Moran, o design da interface de usuário é o design do modelo conceitual a respeito do sistema que o usuário constrói na sua mente (*user's system conceptual model)* [Moran 81]. Moran sugeriu a *Command Language Grammar* (CLG) como uma maneira de estruturar e entender a interface usuário-sistema do ponto de vista conceitual.

A CLG descreve a estrutura da interface de usuário como possuindo três componentes, cada um com dois níveis: *o conceitual, o de comunicação e o físico* [Moran 81]. O componente conceitual possui dois níveis: *o nível de tarefa* e o *nível semântico.* O nível de tarefa tem como propósito estruturar as tarefas de uma maneira que seja adequada ao sistema interativo. O nível semântico compõe-se dos objetos e das manipulações feitas sobre eles. Do ponto de vista dos sistemas eles são estruturas e procedimentos, enquanto que do ponto de vista do usuário estes são entidades e operações conceituais. Neste nível são especificados como as tarefas são compostas em termos de entidades e operações conceituais. O componente de comunicação compreende os níveis *sintático* e de *interação.* O nível sintático determina como o modelo conceitual no nível anterior é organizado numa estrutura lingüística. O nível de interação especifica como as ações físicas são associadas a cada elemento do nível sintático, bem como das regras que governam o diálogo. Este formalismo permite a construção de modelos que se enquadram em duas das categorias apresentada por [Dix et al. 93]: modelos de tarefas e modelos gramaticais.

A divisão do modelo da interação em níveis de tarefa, semântico, sintático, e de interação tem sido largamente adotada em IHC, independente de se utilizar a CLG [Norman 86; Foley & van Dam 82]. A grande vantagem está na separação entre o nível conceitual e o nível das estruturas de interação e em revelar as associações e conexões entre elas, fundamentais para o mapeamento tarefa-ação.

Limitações das Abordagens Cognitivas para a Usabilidade

O principal objetivo da abordagem cognitiva é entendido como o de apresentar um esquema teórico a respeito dos fenômenos de interação com especial ênfase nas atividades cognitivas dos usuários. Entretanto, é preciso entender as limitações destes

modelos e não superestimar o seu potencial. Nesta seção vamos apontar algumas críticas que abrem caminho para as posições assumidas por este trabalho na tentativa de compensar algumas das deficiências apontadas.

Uma limitação importante dos modelos de competência é que na sua maioria eles foram elaborados para modelos de interação em linguagens de comandos. Outra limitação destas arquiteturas ou modelos é o fato de eles enfocarem aspectos cognitivos genéricos dos usuários sem, no entanto, envolvê-los diretamente no processo. Quando isto ocorre, estas técnicas tratam apenas os usuários isoladamente, não abordando aspectos sociais e nem considerando o ambiente de trabalho que os envolve.

Winograd & Flores [Winograd & Flores 86] discutem as limitações da ciência cognitiva e da inteligência artificial em modelar as atividades cognitivas. Os autores propõem que design deve estar fundamentado em comunicação e processos comunicativos, e introduzem uma nova perspectiva baseada na teoria dos atos de fala: a linguagem/ação. O essencial de seu trabalho é ressaltar o papel da comunicação e da linguagem nas atividades cognitivas.

Com o foco nas atividades mentais dos usuários e propondo modelos cognitivos para as atividades de aprendizado, recordação e interpretação, os modelos cognitivos muitas vezes indicam ao designer quais as estruturas mais adequadas para o modelo de interação, como por exemplo qual o vocabulário significativo, como atingir consistência, e assim por diante. Porém, não é proposto ou esclarecido nestas abordagens justamente o fato de que signos e processos de signos são os princípios subjacentes da interação humana com sistemas computacionais [Nadin 88], e, como tal, ao serem produzidos e trocados determinam a eficácia de processos, de uso de estruturas, vocabulários, e assim por diante.

Por exemplo, os modelos de desempenho podem analisar quais operações do sistema precisam ser abstraídas ou refinadas visando diminuir a distância semântica. Da mesma forma, é possível identificar se os comandos devem ser mais ou menos articulados. É papel dos formalismos baseados nestes modelos verificar estes aspectos. Estes modelos permitem ainda identificar problemas que eles mesmo consideram aspectos essenciais sem, no entanto, apontar soluções. Um exemplo é a questão dos

nomes dos comandos que são apontados por várias avaliações experimentais como a causa de diversos problemas. Ela pode ser solucionada através da escolha do melhor signo que represente a operação a ser desempenhada e o seu papel no domínio do usuário. O que precisa ser respondido é quais os significados que um nome de comando tem para um usuário e quais outras lembranças ele traria para o usuário. Como veremos no próximo capítulo, este é um problema para o qual a semiótica aponta caminhos de solução.

A escolha do nível de articulação dos comandos deve considerar as necessidades dos usuários, suas capacidades e limitações. O papel dos modelos de competência está em determinar quais estruturas de comandos devem ser construídas utilizando linguagens de especificação que incorporem modelos cognitivos dos usuários. Estes modelos são essenciais para determinar as características que as regras de interação devem obedecer, como por exemplo, consistência, regularidade e memorização, para minimizar o esforço cognitivo. Estes aspectos não são suficientes para garantir um boa usabilidade do sistema. Mesmo que estes objetivos tenham sido atingidos o usuário continuará necessitando do auxílio dos manuais e dos sistemas de ajuda para saber quais são as regras e as estruturas de comandos da interface. Ainda será preciso ensinar ao usuário qual foi o modelo de interação projetado.

Outra limitação dos modelos cognitivos está em não considerar o real papel que os *menus, rótulos, cabeçalhos e nomes de comandos* desempenham na interface. Eles não fazem parte da estrutura de ações que compõe um comando. Por exemplo, um comando do tipo "*pressione a tecla F12 para salvar"*, compreende apenas a ação de pressionar a tecla. A mensagem associada é apenas um ato comunicativo do *designer* cujo objetivo é orientar o usuário. Da mesma forma, nas linguagens de comandos os nomes dos comandos têm por objetivo esclarecer a sua função. Comandos do tipo "Ctrl+A, Shift+Y" são mais difíceis de ser compreendidos e memorizados, pois os signos escolhidos para representá-los não possuem significado para os usuários.

Nossa avaliação buscou deixar clara a importância e a limitação das soluções apresentadas pelas abordagens cognitivas. Todos estes exemplos levantam problemas de expressão e significação que são do escopo da semiótica. As interfaces gráficas estão

recheadas de elementos com estas funções de significação. É preciso então considerar, dentre as abordagens cognitivas, aquelas que foram desenvolvidas para as interfaces gráficas.

As abordagens cognitivas para interfaces gráficas

O surgimento das interfaces gráficas tem redirecionado algumas das abordagens cognitivas para IHC. Larkin e Simon [Larkin & Simon 87] argumentaram a importância de representações simbólicas no suporte à resolução de problemas e ao desempenho de tarefas em sistemas interativos. Chega-se a sugerir que uma interface de usuário bem projetada elimina a geração e manutenção da estrutura de metas e planos ([Larkin 89], citado em Poitrenaud, e também em [Dix et al. 93], [Howes & Young 96]). Draper [Draper 86] introduz o termo *entrada/saída co-referencial* para ressaltar a importância do *display*[10] no processo de interação como meio de se fazer referências cruzadas no processo de diálogo com o sistema. Comandos expressos através de interfaces gráficas, baseadas no display, permitem que se faça a *co-referência* às mensagens previamente veiculadas pelo sistema e pelo próprio usuário.

O interesse no display tem sido uma resposta aos modelos cognitivos formais que se mostraram insuficientes para refletir o contraste entre a facilidade de aprendizado em interfaces baseadas em caracteres e as interfaces WIMP baseadas no display.

O salto teórico foi dado ao se tentar elaborar modelos mais genéricos que sejam capazes de explicar como o usuário pode construir modelos mentais motivados pelas representações no display. A idéia básica é que usuários devem adquirir o modelo mental/cognitivo a respeito do sistema através das imagens e mensagens mostradas no display.

Um exemplo desta mudança de postura está nos trabalhos de Norman. Embora em um trabalho anterior ele tenha enfatizado a importância de modelos do conhecimento que o usuário tem do sistema [Norman 83], na engenharia cognitiva ele ressalta a importância da *imagem do sistema* na maneira como este modelo é formulado na mente do usuário [Norman 86]. Mais tarde ele deixa evidente sua mudança de postura ao

[10] A tela gráfica do computador.

considerar a importância do papel das representações na *imagem do sistema* como fundamental para se atravessar os golfo da interação [Norman 93].

Nas abordagens baseadas em display, o modelo de Norman deve ser visto de forma invertida [Kitajima & Polson 95]. O *ciclo de ação* deve começar a ser percorrido pelo golfo da avaliação, onde os símbolos do display precisam ser interpretados e avaliados com as possíveis metas que o usuário quer executar. Estes símbolos são também a motivação para a travessia do golfo da execução. Neste caso a tarefa cognitiva é selecionar do display os possíveis candidatos à formulação e especificação das ações.

O TAL *(Task-Action Learner)* é um modelo de desempenho compatível com observações empíricas de aprendizado humano. Ele permite ao designer avaliar o desempenho do usuário no aprendizado de diferentes interfaces. O modelo TAL revela a importância da interatividade através do display como um dos fatores importantes no aprendizado [Howes & Young 96]. A ênfase está em resolver o problema do mapeamento tarefa-ação através da *interatividade.*

Uma evolução das TAGs, as D-TAGs consideram a possibilidade de revelar o modelo de competência no *display* [Howes & Payne 90]. As D-TAGs incorporam o conhecimento a respeito do display com o conhecimento a respeito da tarefa de maneira a determinar a tarefa associada. O estado do display incentiva as ações possíveis e restringe as ações não válidas num determinado momento.

Um exemplo de aplicação do display a interfaces em linguagem natural pode ser visto nas interfaces em *linguagem natural orientada por menus* [Calejo et. al. 86; Leite 91; Garcia 95, Tennant 83]. Neste estilo de interface, menus no display orientam ~~a~~o usuário na formulação de *todas e apenas* as sentenças em linguagem natural que podem ser interpretadas por uma aplicação. Neste caso o que está sendo mostrado aos usuários são as restrições da competência do sistema em relação à (mais poderosa) competência do usuário na sua própria língua.

O que caracteriza todos estes trabalhos é considerar o papel das representações através de símbolos, gráficos e linguagens como de fundamental importância na interpretação do modelo de interação. O display passa a ter uma posição importantíssima na interação usuário-sistema, tanto no desempenho (ver [Kitajima & Polson] 95) quanto

na competência do usuário ([Payne 89],[Howes & Payne 90]). É importante ressaltar que nesta linha de abordagem o foco não está mais totalmente concentrado em aspectos cognitivos do usuário. Ampliou-se o foco e passou-se a considerar a importância do display como tendo um papel (de ferramenta) importante na cognição do usuário. Entretanto, a perspectiva adotada ainda é a de construir modelos cognitivos de como usuários interpretam e interagem com o sistema a partir das representações no display.

A ênfase na representação em interfaces baseadas no display apresenta uma alternativa interessante que facilita o problema do *aprendizado e recordação* do mapeamento tarefa-ação. Entretanto isoladamente esta solução não resolve por completo o problema do mapeamento

Em todas estas propostas a questão latente e não resolvida é de como as mensagens surgem no display, de que agente intencional se originam, e em que contexto comunicativo devem ser interpretadas globalmente. Em todas as soluções aqui apresentadas os signos do display nunca são explicitamente apresentados como uma mensagem cuidadosa e intencionalmente elaborada por um designer e enviada para os usuários de seu artefato de software através do conjunto de recursos disponíveis nas interfaces gráficas. Esta ausência do personagem mentor e emissor legítimo de todas as mensagens de interface gera potencialmente uma importante distorção interpretativa na mente do usuário que pode vir a formar um modelo mental em que a comunicação se inicie e se encerre no "sistema", então entendido como um agente intencional autônomo. Isto só pode acarretar inúmeras e muito importantes falhas de expectativa para o usuário, de vez que uma perspectiva antropomorfizante é virtualmente insustentável para qualquer artefato de software. Portanto, o problema com as abordagens cognitivas para interfaces baseadas no display é considerar que tudo ocorre apenas no eixo que se estende entre o usuário e a interface, desconsiderando o papel do designer e do contexto de uso na elaboração de representações. O autor legítimo das informações ou mensagens é justamente o designer. Este aspecto é a premissa básica da Engenharia Semiótica.

As Abordagens Semióticas

A Semiótica é a disciplina que estuda os signos, os sistemas semióticos e de comunicação, bem como os processo envolvidos na produção e interpretação de signos.

Existem diversas propostas e definições para semiótica, bem como para signos e demais fenômenos envolvidos.

Existem diversas tradições em Semiótica. As mais citadas na literatura incluem a de Charles S. Peirce [Peirce 31], Ferdinand de Saussure [Saussure 16] e Louis Hjelmslev [Hjelmslev 63]. A semiótica de Peirce destaca a importância do papel dos signos no raciocínio humano. Para Peirce semiótica é *lógica*. A *semiologia* de Saussure está fortemente enraizada na *lingüística*. Louis Hjelmslev desenvolveu a partir da teoria de Saussure uma teoria chamada *glossemática*, que permite uma análise mais específica, detalhada e formal das estruturas dos signos lingüísticos. Mais recentemente Umberto Eco [Eco 76] apresentou a sua teoria de semiótica estendendo conceitos das tradições de Peirce e Hjemlslev aplicando-os às artes e cultura em geral.

A Semiótica é tradicionalmente dividida em três campos de estudos [Andersen et al. 93]:

- O signo, os diferentes tipos de signos e os diferentes processos através dos quais eles adquirem e transmitem significado;
- Os códigos ou sistemas que organizam signos, incluindo comunicação, codificação e decodificação.
- A cultura na qual os signos são utilizados.

A semiótica em si não pode ser considerada uma teoria completa sobre os fenômenos de significação e comunicação humanas. A semiótica é ainda um campo de estudos no qual existe diversas correntes. Dentro destas correntes se apresentam diversos modelos teóricos. O que temos a questionar a respeito destas abordagens é quais são as suas reais vantagens sobre outras abordagens teóricas, avaliar se elas são necessárias e produzem resultados satisfatórios e quais dentre as teorias semióticas podem ser aplicadas ao desenvolvimento de sistemas interativos.

As abordagens semióticas para IHC caracterizam- se principalmente pela aplicação de teoria semiótica como fundamentação teórica para explicações de fenômenos relativos à interação usuário-sistema e à sua aplicação no design de interfaces de usuários. Sistemas são observados como máquinas que produzem e processam

elementos de signos, enquanto que os usuários são participantes dos processos que os definem.

O foco de interesse não está diretamente centrado em atividades mentais dos usuários, mas no papel que sistemas de significação e comunicação exercem sobre estas atividades cognitivas. Ao adotar a semiótica como uma fundamentação para IHC pode-se trazer não apenas novos conceitos e teorias, mas diferentes tradições e perspectivas filosóficas.

As motivações para a aplicação de semiótica em IHC são as mais variadas. A natureza simbólica de computadores e do processo de interação, associada às limitações da lingüística para lidar com interfaces gráficas, é uma das mais citadas. A disseminação de aplicações computacionais como ferramenta de comunicação ou *medium* também tem chamado a atenção para a elaboração de sistemas de comunicação através de computadores. Semioticistas têm demostrado interesse em análise semiótica de aplicações no *medium* computacional. Aplicações multimídia também são construídas utilizando-se técnicas de comunicação. Outra motivação é a limitação das abordagens cognitivas para a prática de design, discutidas na seção anterior, bem como as vantagens das interfaces gráficas em informar ao usuários com respeito ao modelo de interação e funcionalidade do sistema. Entretanto, a motivação maior de nossa abordagem está na proposta de Andersen, citada em [Kammersgaard 88], na qual uma perspectiva particular de se observar sistemas como *medium* ocorre no caso em que o projetista se comunica com o usuário. Esta perspectiva segue a já citada perspectiva de Alan Kay de considerar sistemas como *meta-medium* [Kay 77].

Existem outras aplicações de semiótica em computação como por exemplo, em *linguagens de programação* no pioneiro trabalho de [Zemanek 66], em *análise de requisitos* e *engenharia de software* [Leite & Franco 93; Pimenta & Faust 97], na engenharia de informações, métodos e organizações [Stamper 92]. A estas diversas aplicações em análise e desenvolvimento de sistemas deu-se a denominação de *Semiótica Computacional* [Andersen 90]. Por ser um campo de pesquisa ainda em desenvolvimento alguns autores utilizam o termo semiótica computacional com outros propósitos, como teoria de sistemas, inteligência computacional e cibernética [Albus & Meystel 95].

A Semiótica Computacional de Andersen

Andersen foi um dos primeiros pesquisadores a introduzir fundamentos de teoria semiótica em computação propondo uma *Teoria da Semiótica Computacional* [Andersen 90]. Andersen se baseia na semiótica estruturalista (a glossemática) de Hjelmslev e seu trabalho se caracteriza por apresentar uma análise detalhada dos signos computacionais. Mais recentemente, Andersen aplica a semiótica de Peirce para o estudo da utilização de computadores e também propõe um framework teórico para programação.

A visão de Andersen é de que sistemas computacionais devem ser vistos como *medium* ou sistema semiótico. Ele se refere a *medium* não no sentido comum de rádio ou televisão, mas um meio extremamente flexível e polimórfico. A propósito, foi a partir da visão de Andersen que se passou a considerar a perspectiva de que projetistas são emissores de mensagens através do *medium* computacional.

A abordagem de Andersen é analítica e estruturalista. Ele propõe taxonomias que revelam a natureza do *medium* e dos signos computacionais e apresenta uma descrição das estruturas que formam estes signos. Sistemas computacionais são analisados por estruturas estáticas e dinâmicas. A descrição dos signos em diagramas de redes de Petri, já utilizada em design e modelagem de diálogos, não acrescenta muita coisa à área. Ela se diferencia da abordagem de ciência da computação tradicional pela inclusão dos conceitos estruturalistas da lingüística de forma/substância, expressão/conteúdo, paradigma/sintagma e concorrência/seqüência, o que revela uma nova perspectiva e permite uma nova interpretação dos modelos computacionais. O trabalho de Andersen é marcado pela transferência e analogia de técnicas e métodos da análise semiótica de linguagens e artes cenográficas para o domínio computacional. Interfaces e interação são estruturadas como cenografias, enquanto que programação é estruturada como linguagem. A ênfase na estrutura semiótica dos signos, embora bastante útil por possibilitar novas perspectivas, não se apresenta muito diferente das estruturas lingüísticas já utilizadas em linguagens de programação e em interfaces e hipertextos.

A teoria proposta por Andersen apresenta vantagens para análise de sistemas computacionais, entretanto não discute aspectos de produção e interpretação dos signos o que torna a sua aplicação no design de interfaces e aplicações bastante limitada. O design

é considerado como a composição de estruturas de signos que devem determinar os padrões de uso.

O Paradigma Semiótico para Design de Interfaces de Nadin

A proposta de Nadin [Nadin 88] se mostra bastante interessante por enfatizar a natureza semiótica da atividade de design, argumentando especificamente que o design de interfaces de usuário deve estar baseado na lógica semiótica de Peirce. Nadin enfatiza uma mudança de paradigma de design e argumenta ainda que interfaces possuem a natureza representativa de signos e que o seu design e utilização são processos de natureza semiótica. Desta forma, semiótica como –teoria e prática dos processos semióticos deve considerada uma fonte de conhecimentos para o design de interfaces de usuários.

O trabalho de Nadin justifica exaustivamente as abordagens semióticas para design de maneira a nos dispensar de maiores argumentações em prol da aplicação de teoria semiótica para o design. Sumariamente, ele enfatiza que se existe uma ciência de interfaces (computacionais ou de qualquer outro tipo), então esta ciência é a semiótica, e a lógica semiótica de Peirce parece ser apropriada para interfaces (*op. cit.*, pag. 272). Isto explica, segundo ele, porque o design de interfaces não se apresenta como um processo de engenharia baseado em modelos teóricos e possui características de processo de produção artística.

A premissa para se considerar a interface de usuário a partir de um ponto-de-vista semiótico é a de que ela representa um sistema de signos complexo, a *linguagem de interface*, que representa a sua aplicação (*op. cit.*, pag. 275). Nadin apresenta um método para design e um modelo para interfaces. O método considera aspectos pragmáticos que relacionam a linguagem com a aplicação que ela representa. Esta linguagem, constituída por um repertório e regras de uso, determina protocolos de baixo nível que constituem o ambiente, as ferramentas, as atividades e os suprimentos da aplicação representados pelos signos da interface.

A abordagem de Nadin se diferencia da de Andersen por enfatizar mais a importância de um design pragmático e menos a estrutura do sistema semiótico da interface. Nadin considera que a comunicação é a atividade semiótica que coloca usuários

e designers juntos. Ele insiste ainda que a linguagem usada nesta comunicação deve ser, em princípio, formal. Nestes aspectos o trabalho de Nadin se assemelha com a nossa proposta de Engenharia Semiótica. Entretanto, a metodologia de Nadin limita-se a representações de elementos da interface considerados dentro do contexto. Ele considera que o *tipo* e a *quantidade* de signos são decisões de design importantes que afetam representações. Estes fatores são de fato importantes, mas não se considera o aspecto dinâmico e computável dos signos envolvidos. Além disto, não se considera tampouco como os signos formam um modelo de interação global. O que está se representando, como sabemos, não são apenas objetos da aplicação, mas computações sobre estes objetos e por esta razão motivamo-nos a explorar os limites desta proposta.

Nadin também não apresenta nenhum modelo mais detalhado a respeito da interface e de suas estruturas, nem como isto pode ser mapeado em modelos computacionais. A linguagem da interface (do sistema de signos) é apresentada de maneira informal e não estruturada e resta saber como estes aspectos semióticos podem se tornar métodos práticos para o design de interfaces.

Outros

Nake [Nake 94] considera que a interação usuário-sistema é constituída por dois processos acoplados: um processo de signo e um processo de sinal (*a sign process* and a *signal process*). O usuário está envolvido no primeiro e o sistema no segundo. O processo de sinal é determinístico, governado pelos modelos e algoritmos formais do sistema, enquanto que o processo de signo permite que o usuário o utilize criativamente e tenha várias interpretações a respeito do processo de sinais. Matemática e informática são a base teórica para os processo de sinais, enquanto a semiótica pode oferecer teoria para os processos de signos

Os processos de sinais são a expressão (a parte física) que forma os signos nos quais o usuário está envolvido. Nake coloca o problema de design de sistemas interativos usáveis como o design do processo de sinais que, quando ativados, possibilitam ao usuário associar interpretantes significativos e pretendidos para os sinais de entrada e saída, dentro do seu próprio contexto e situação.

No referido trabalho, Nake apresenta excelentes justificativas para a aplicação de semiótica e uma perspectiva para o processo de interação que revela distinções importantes. Entretanto, ele não apresenta maiores detalhes destes processos semióticos nem sobre a sua aplicação prática no design, exceto pela argumentação de que design de interfaces é a construção de significados a partir da manipulação de sinais pelo computador e que gera processos de signos no usuário dentro do seu contexto de uso.

Existem outras propostas de aplicação semiótica para design de interfaces. Mullet e Sano [Mullet & Sano 95], num trabalho sobre design gráfico do layout de telas de interfaces, colocam a semiótica como a base para representações e imagens no design. Eles discutem alguns do conceitos de Peirce, entretanto, não fica claro qual a aplicação destes conceitos nos princípios e técnicas do design. Marcus [Marcus 92], num trabalho também em design gráfico de telas, apresenta diretrizes para a elaboração de ícones ressaltando a clássica taxonomia de Peirce de signos. Ambos os trabalhos, embora com grande aplicação para a prática de design, utilizam conceitos de semiótica superficialmente na elaboração de diretrizes de projetos, sem aprofundar-se nas questões teóricas, tal como os trabalhos mencionados acima. Além disto, o trabalho deles limita-se aos aspectos estáticos da interface, comuns a outras formas de comunicação gráfica, e não aborda os aspectos dinâmicos e computáveis da interface.

Considerações Finais

Nesta seção vamos apresentar um resumo tabular da nossa avaliação das principais abordagens para design de interfaces de usuário, descrevendo resumidamente como cada uma satisfaz aos critérios estabelecido no início do capítulo (ver Tabela 3-1).

Todos estes trabalhos mostram o quanto teorias das ciências humanas têm muito a contribuir para o estudo do processo de interação usuário-sistema e especialmente para a atividade de design. O que fica evidente em todas as abordagens é que o design de interfaces não pode ser considerado a partir de uma perspectiva centrada-no-sistema ou centrada-no-usuário apenas. Em nossa abordagem ressaltamos a importância dos fenômenos de significação e do contexto e situação de uso na qual se encontram os usuários como fundamentais para a interação.

Isto indica que a semiótica, como disciplina que estuda o processo de semiose que envolve atividades mentais e comunicação dentro de uma postura pragmática, pode ser tomada como um esquema teórico geral para o domínio conceitual da IHC que complementa as abordagens cognitivas.

As abordagens semióticas estudadas mostram-se limitadas em alguns aspectos. Nosso trabalho apresenta alternativas para algumas destas limitações.

Categoria da abordagem	Sub-categoria da abordagem	Fontes teóricas	Modelos teóricos e perspectivas	Contribuições para o design	Principais vantagens e limitações
Abordagens cognitivas	Avaliação empírica	Psicologia cognitiva		Métodos analíticos para avaliação	Permitem identificar problemas no design, mas não apontam as soluções
	Modelos cognitivos de performance e competência	Psicologia cognitiva e lingüística, especialmente: [Chomsky 57], [Simon & Newell 72], Card, Moran & Newell, 83] e [Norman 86]	A interação é um processo de mapeamento tarefa-ação realizado por processos cognitivos do processamento de informação.	Modelos dos processo cognitivo, e formalismos (gramáticas de ação) para o design de interfaces.	Conseguem fazer previsões e explicações para alguns aspectos de desempenho do usuário. Não consideram os fenômenos de significação e comunicação.
	Intefaces baseadas no display	Psicologia cognitiva	A interação é um processo de mapeamento tarefa-ação auxiliado pelo display que minimiza o esforço cognitivo.	Extensões dos modelos cognitivos visando considerar o papel do display e das representações.	Abordam problemas de significação sem no entando considerar o papel fundamental do designer.
Abordagens Semióticas	Teoria de Semiótica Computacional: Andersen [91]	Semiótica estruturalista de Hjelmslev	A interface é um sistema estruturado de signos.	Modelo do sistema computacional utilizando os principais conceitos do estruturalismo.	Apresenta uma ótima análise dos sistemas de signos, do sistema computacional e do domínio, sem no entanto revelar como aplicá-los
	Paradigma Semiótico Nadin [88]	Semiótica de Peirce	A inteface é um sistema lógico-semiótico.	Um metodologia para a escolha dos signos.	Não apresenta métodos ou ferramentas precisas para o design da interface.
	Nake [94]	Semiótica	A interface e o usuário compõem um processo de signo.	Não apresenta contribuições significativas.	

Tabela 3-1: Resumo das abordagens teóricas.

A ENGENHARIA SEMIÓTICA DE INTERFACES DE USUÁRIO

Neste capítulo começaremos a apresentar a nossa abordagem para design de interfaces de usuário. A Engenharia Semiótica, proposta por [de Souza 93][11], é uma nova abordagem fundamentada em teoria semiótica. Neste capítulo apresentaremos a perspectiva teórica lançada naquele trabalho, bem como algumas extensões. Para melhor compreendê-la apresentaremos alguns conceitos fundamentais de semiótica. Em seguida ilustraremos os conceitos com um estudo de casos. Finalmente, discutiremos as implicações da perspectiva teórica no desenvolvimento de software e os requisitos para o design de interfaces de usuário.

A abordagem da Engenharia Semiótica

A abordagem da Engenharia Semiótica apresenta uma perspectiva para IHC na qual o sistema computacional é um artefato de *metacomunicação* e através dele o designer envia uma mensagem um mensagem para os usuários cujo conteúdo deve ser o modelo de interação e de funcionalidade do sistema [de Souza 93]. O conteúdo de tal mensagem é o modelo de usabilidade da aplicação. A sua expressão é formada pelo conjunto de todas as mensagens de interação veiculadas na interface durante o processo de interação. O usuário exerce o duplo papel de interagir com o sistema e interpretar uma mensagem enviada pelo designer. A Engenharia Semiótica está fundamentada nas teorias semióticas propostas por *Charles S. Peirce e Umberto Eco*.

A Engenharia Semiótica surgiu com o objetivo de subsidiar soluções práticas para o design de interfaces de usuário ainda não alcançadas por outras abordagens,

[11]Há uma proposta simultânea de René Jorna, usando também a expressão *Engenharia Semiótica*, mas numa acepção bem diferente, que se aproxima muito mais de Semiótica Organizacional do que de uma pretendida Teoria de Design de Interfaces [Jorna 93; Jorna & van Heusden 96].

especialmente as abordagens cognitivas. Ela tem suas origens em idéias provenientes de vários autores. A primeira delas, apresentada por Kammersgaard, considera que sistemas computacionais podem ser vistos como *medium*, e que um caso especial desta perspectiva são as aplicações de software que colocam o designer na posição de emissor através do *medium* computacional [Kammersgaard 88]. A segunda delas considera que a usabilidade de sistemas deve ser vista como a qualidade que permite a usuários utilizar aplicações como ferramenta intelectual para resolver problemas explorando suas capacidades cognitivas [Adler & Winograd 92]. A terceira é que a semiótica pode proporcionar os fundamentos teóricos para a compreensão de fenômenos da IHC, especialmente para a atividade de design [Nadin 88, Andersen 90].

A partir destas idéias, a Engenharia Semiótica argumenta que o desafio de usabilidade pode ser resolvido com a perspectiva de que designers devem projetar e comunicar uma aplicação de software cuja interface seja vista como um sistema de comunicação para o usuário interagir e como um *medium* que (meta)comunica[12] sua funcionalidade e o próprio sistema de comunicação [de Souza 93]. Para colocarmos em prática esta perspectiva, é preciso considerar uma hipótese semiótica fundamental: a de que signos e sistemas semióticos são a ferramenta intelectual mais poderosa que as pessoas têm para a aquisição e comunicação de conhecimento [Peirce 31] e que sistemas computacionais são tipicamente máquinas semióticas, i.e., representam conceitos e processos do conhecimento humano [Eco 88].

A idéia original da Engenharia Semiótica estava limitada ao design de interfaces de usuário. Recentemente ela também vem se mostrando útil como um framework mais genérico para o design de *linguagens para usuários* [de Souza 97]. Diversos trabalhos conduzidos pelo SERG[13] estão ampliando a aplicação da Engenharia Semiótica em *linguagens de programação para usuário final, compreensão de programas, interfaces multi-usuário e multi-modais, interfaces e linguagens para bases de conhecimento*.

[12]Metacomunica, i.e., comunica algo sobre ou através do próprio processo de comunicação, tal qual, por exemplo, a personalidade de alguém se revela através do modo como este alguém conversa com os outros.
[13]*Semiotic Engineering Research Group* da PUC-Rio, do qual o autor desta tese faz parte.

A Perspectiva de Metacomunicação para IHC

As interfaces de usuários são artefatos de metacomunicação, uma vez que são não apenas um sistema de comunicação usuário-aplicação, mas também a mídia por onde se comunica para o usuário uma solução de design - funcionalidade e modelo de interação - formulada pelo projetista. O que ele transmite não é uma mensagem como a de um documento multimídia, um livro, ou um filme, mas uma mensagem interativa e performática: um sistema de comunicação (para a interação) e um resolvedor de problemas (a funcionalidade da aplicação). Visto por esta perspectiva, o design de interfaces envolve não apenas a concepção do modelo de interação, mas a *comunicação* deste modelo da aplicação de maneira a revelar para o usuário o espectro de usabilidade da aplicação. Esta perspectiva está ilustrada na Figura 0-1.

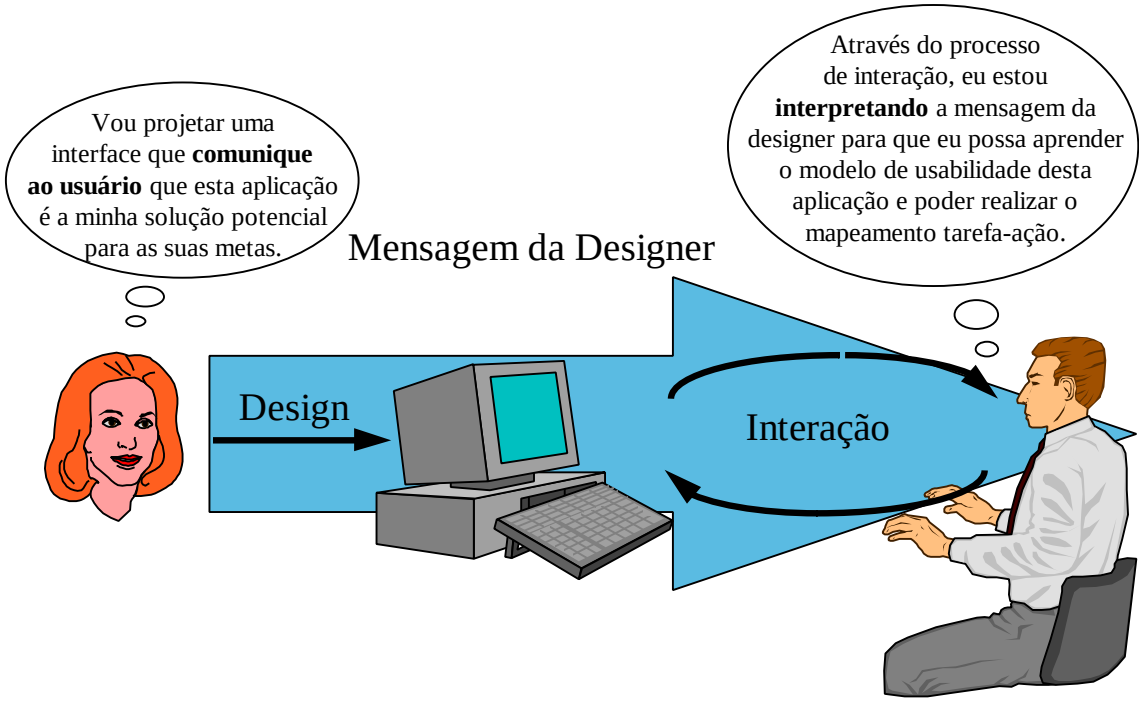


Figura 0-1: A perspectiva de metacomunicação da Engenharia Semiótica

Na Engenharia Semiótica o foco está na comunicação interpessoal entre o designer e os usuários. Sistemas computacionais oferecem meios de se criar um artefato *ativo* onde o modelo de interação e a funcionalidade da aplicação podem ser ensinados aos usuários. Manuais, sistemas de ajuda online, tutoriais, seções de treinamento são exemplos de ferramentas de apoio neste processo de ensino/aprendizado. O argumento

principal da Engenharia Semiótica é que o modelo de interação e a funcionalidade da aplicação devem ser parte desta mensagem unidirecional e indireta enviada através da coleção de signos da interface. O projetista passa a ter um papel comunicativo explícito ao se utilizar da interface para dizer algo ao usuário. Este é um processo demonstrativo, no qual o designer está dizendo: *"Veja o que eu estou querendo dizer!"*[14] O usuário tem um duplo papel: o de *agente da interação* e o de *receptor na comunicação com o projetista.*

A abordagem da Engenharia Semiótica, portanto, não se aplica apenas a interfaces gráficas, uma vez que a mensagem não precisa ser comunicada necessariamente por um display gráfico. A perspectiva e os modelos elaborados em nossa abordagem podem ser aplicados para interfaces quaisquer, conforme veremos em exemplos a seguir.

A Mensagem do Designer

Quando o usuário entra em contato visual (ou, mais genericamente, sensorial) com a interface, ele realiza um esforço de interpretação e compreensão a respeito do significado de todos os seus dispositivos e da informação que eles veiculam. Por exemplo, o usuário, quando observa um widget na interface, precisa saber como ele pode manipulá-lo e qual será o comportamento do sistema após a sua ação. O que está sendo interpretado pelo usuário, mesmo que despercebidamente, é o que o designer quis dizer sobre aquele widget. A interpretação que a Engenharia Semiótica oferece para este processo é a de que a interface está transmitindo uma mensagem para o usuário, do tipo *"Eis aqui um botão. Aperte-o e ele realizará 'tal' operação"*. Este estilo de mensagem é referido como *affordance* [Gibson 79]. A interface, isto é, os diversos elementos que a formam (mouse, teclado, widgets, menus, caixas de diálogo, instruções, avisos, comandos, etc) é, pois, uma mensagem formada por outras mensagens.

Uma característica da mensagem do designer é que ela é dinâmica. Ela pode variar ao longo do tempo, revelando variações de significado. Este é o caso dos objetos de interfaces indicadores de estado ou comportamento de um sistema. Mensagens desse tipo se comportam como agentes ativos e são chamadas de *performáticas.* Outro caso de mensagem dinâmica é quando ela pode ser construída interativamente pelo usuário. As

[14] *Watch what I mean!*, numa referência a *Watch what I do!* [Cypher 93]

mensagens *interativas* devem possibilitar e incentivar ações do usuário. Um exemplo são as cascatas de menus que vão sendo construídas durante o processo de interação. Vale ressaltar que esta característica ocorre apenas nos objetos de interfaces virtuais, veiculados no vídeo.

Vejamos, como exemplo, um programa de subtração bem simples. Suponhamos o seguinte processo de interação (os signos de interação serão rotulados para referências) entre o sistema (S) e o usuário (U):

S: *"Forneça o primeiro número:"* (I)

U: *"5.3"* (a).

S: *"Você deveria fornecer apenas números inteiros, em notação decimal".* (II)

U: *"5"* (b).

S: *"Forneça o segundo número:"* (III)

U: *"8"* (c).

S: *"O resultado da subtração é* (IV) *: -3* (d)*"*

O sistema apresenta as mensagens (I), (II), (III), (IV) e (d) na tela do computador e o usuário fornece as representações numéricas em (a), (b) e (c) usando o teclado do sistema. Todas estas *mensagens da interação,* tanto signos do sistema quanto do usuário, fazem parte da *Mensagem do Designer.*

O modelo de usabilidade inclui a funcionalidade que se refere ao processo de subtrair dois números inteiros enquanto que o modelo de interação determina quais o signos que o usuário deve utilizar, como deve produzir estes signos na interface e como interpretar os resultados da aplicação. O modelo de usabilidade é implementado, e conseqüentemente deve poder ser representado formalmente, pela linguagem de programação na qual é codificado. Porém, com as representações utilizadas na interface, o usuário pode adquirir um modelo de usabilidade diferente, como passaremos a mostrar.

Ao interagir com o sistema o usuário pode interpretar o significado da mensagem fornecendo números inteiros na notação decimal e interpretando o signo resultante. Ele estará assim adquirindo o modelo de usabilidade da aplicação. As mensagens (I), (II),

(III) e (IV) são mensagens do designer veiculadas através da interface. Elas são invariantes em todos os usos da aplicação. Os signos (a), (b) e (c) são as mensagens do usuário para o sistema, enquanto que o signo (d) é uma mensagem do próprio sistema para o usuário. Elas variam em cada caso de uso.

Suponha que a mensagem (IV) fosse apenas *"O resultado é:"*. Neste caso é ainda mais difícil para o usuário atribuir um significado a respeito da funcionalidade do sistema. Toda interpretação correta dependerá exclusivamente do conhecimento prévio sobre a funcionalidade da aplicação, uma vez que em nenhuma mensagem do designer para o usuário figura a palavra equivalente ao conceito central neste cenário: a subtração. Isto exemplifica o quanto as mensagens escolhidas pelo designer facilitam ou não a interpretação da funcionalidade do sistema. Em um outro caso, suponha que a mensagem (I) fosse *"Forneça o primeiro número inteiro"*. Neste caso, o usuário tem condições de saber mais ainda sobre a funcionalidade do sistema e também sobre o modelo de interação, pois assim ele sabe que tipo de representação ele pode utilizar com este sistema. Assim, seria possível evitar o erro avisado pela mensagem (II).

Em qualquer das mensagens não é dito nada a respeito da ação física que o usuário deve realizar para fornecer os números. A decisão do designer, supondo que os usuários normalmente fornecem dados numéricos teclando símbolos e acrescentando <ENTER> ao final, foi de não veicular mensagens através do sistema a respeito destes detalhes de interação. Para usuários mais inexperientes a mensagem (I) poderia ser *"Forneça o primeiro número inteiro pressionando as teclas numéricas correspondentes e depois pressionando a tecla* <ENTER>*"*. O designer claramente revela com isto sua pressuposição a respeito do conhecimento que o usuário já possui sobre o modelo de interação.

Numa interface radicalmente diferente onde as mensagens (I), (III) e (IV) fossem substituídas por um signo mais protocolar como um ">", a funcionalidade do sistema e o sistema de acionamento continuariam sendo os mesmos. A diferença estaria em que o designer não teria utilizado-se da interface para comunicar a usabilidade da aplicação para o usuário.

Existem muitas outras alternativas para estas mensagens do designer. As mensagens (I) e (III) poderiam ser, respectivamente, *"Forneça o diminuendo"* e *"Forneça o subtraendo"*, de maneira a comunicar mais especificamente, utilizando termos do domínio da aplicação, a funcionalidade disponibilizada. Esta opção pode, porém, dificultar a interpretação de usuários não muito lembrados de sua aritmética.

As alternativas apresentadas acima não alteram a funcionalidade do sistema nem o modelo de interação. Elas são alternativas de mensagens do designer para o mesmo modelo de usabilidade do sistema. Entretanto, com cada alternativa os usuários adquirem diferentes *modelos conceituais de usabilidade*. Ou seja, estas diferentes interpretações das mensagens da interface são responsáveis por diferentes aproveitamentos do sistema. A Engenharia Semiótica considera a usabilidade do sistema como função do modelo que é comunicado pelo designer através da interface.

Um outro dado importante é que as mensagens (I), (II), (III) e (IV) são formuladas em tempo de design, mas só são apresentadas em tempo de execução. A mensagem (II), a propósito, apenas é comunicada aos usuários em caso de erro, ou seja, ela é situada e é opcional no curso da interação, dependendo de um equívoco sobre o protocolo de entrada.

Neste exemplo bastante simples o designer utilizou diferentes sistemas semióticos (códigos) nas mensagens da interface. A língua portuguesa foi utilizada para a comunicação do designer com o projetista sobre o modelo de usabilidade nas mensagens (I), (II), (III) e (IV). O sistema numérico decimal foi utilizado para a codificação das quantidades a serem subtraídas nas mensagens produzidas pelo usuário e pela aplicação. Já nas interfaces gráficas estilo WIMP (GUI/WIMP), o designer poderia se utilizar de elementos gráficos. Por exemplo, usando um widget tal como um botão de comando contendo no seu interior o *sinal de igualdade* "=", o designer está enviando diversas mensagens. O ícone do botão expressa algo que pode ser interpretado pelos usuários como *"Clique aqui"* se o usuário conhecer o *código de botões*. Esta mensagem diz respeito a como o usuário deverá interagir, ou seja o referente do signo empregado está no modelo de interação. O rótulo deste botão (o sinal de igualdade "=") expressa algo sobre a funcionalidade da aplicação.

Um mesmo signo de interface pode, porém, cumprir objetivos distintos em mais de uma direção e sentido de comunicação. Por exemplo, além desta função comunicativa na mensagem do designer para o usuário, o widget pode veicular mensagens do usuário para a aplicação, como quando este clica sobre o botão com o mouse. Adicionalmente, o mesmo widget pode veicular mensagens da aplicação para o usuário indicando algo a respeito do estado do sistema.

Embora estes exemplos sejam relativos a um nível de granularidade bastante pequeno, eles dão a diretriz principal de nossa abordagem. Numa aplicação de software típica, a funcionalidade e o modelo de interação são muito mais complexos. Todas estas mensagens devem utilizar interpretadores e geradores mais sofisticados. O ponto de vista ressaltado na Engenharia Semiótica [de Souza 93] é considerar que o designer é autor desta mensagem, que sua presença no cenário comunicativo deve ser explicitada e tornada sensível para os usuários, e que sua mensagem é transmitida pelo processo de interação que caracteriza o processo metacomunicativo.

Para entender-se a fundo todas as implicações que a perspectiva aqui introduzida traz para o design de interfaces é preciso compreender alguns conceitos fundamentais de semiótica. Estes conceitos permitirão definir em detalhes os requisitos necessários, isto é, a identificação de problemas a serem resolvidos a partir desta nova perspectiva.

Fundamentos de Semiótica: Signo e Comunicação

A semiótica é disciplina que estuda os signos, sistemas de signos, significação, comunicação e todos os processos culturais [Eco 76]. Existem diversas definições e considerações epistemológicas a respeito de semiótica que não serão discutidas neste trabalho. Ficaremos restritos à tradição de Peirce [Peirce 31] e ao tratado proposto por Eco [Eco 76]. Na Engenharia Semiótica é fundamental entender a noção de signo como colocada por Peirce e a teoria de produção de signos proposta por Eco.

Um dos motivos principais para a escolha das semióticas de Peirce e Eco é que elas oferecem uma postura pragmática e uma oportunidade de considerar a cultura e os instrumentos culturais como a base para as atividades intelectuais. Não existe uma preocupação com os processos mentais, mas com o papel dos sistemas no conhecimento e comportamento humano.

Um ***signo*** é definido por Peirce como:

> "[...] aquilo que, sob certo aspecto ou modo, representa algo para alguém. Dirige-se a alguém, isto é, cria na mente dessa pessoa um signo equivalente, ou talvez um signo mais desenvolvido. Ao signo assim criado denomino *interpretante* do primeiro signo". [Op. Cit. (pag. 46)][15]

> "Qualquer coisa que conduz uma outra coisa (seu *interpretante*) a referir-se a um objeto ao qual ela mesma se refere (seu *objeto*), de modo idêntico, transformando-se o interpretante, por sua vez, em signo, e assim sucessivamente *ad infinitum*" [Op. Cit. (pag. 74)]

O signo não deve ser visto como uma entidade ou elemento, mas como um *processo.* Algo que requer a presença de alguém e que na mente desta pessoa desencadeia interpretantes, num processo ilimitado ao qual Peirce denominou *semiose ilimitada.* As conseqüências destes conceitos são fundamentais na Engenharia Semiótica. São eles que avalizam a perspectiva de metacomunicação e que diferenciam a nossa abordagem das abordagens cognitivas. A noção de modelo mental, fundamental nas abordagens cognitivas, deveria (embora não o faça de maneira explícita e sistemática) considerar este aspecto ilimitado da semiose.

O signo é produto de uma relação entre três elementos: a *expressão (representamen)*[16]- que é aquilo que representa; o *objeto* - aquilo que é representado; e o seu *interpretante* - a capacidade mental no contexto do processo do signo [Peirce 31]. O relacionamento entre estes três elementos está na Figura 0-2.

No signo, o interpretante não deve ser confundido com intérprete. "Quando nós interpretamos um signo, nós nos tornamos parte dele no instante da interpretação" [Nake 94]. Esta atividade mental, no conceito de Peirce, não está limitada ao processo de interpretação estático, mas a toda atividade e associação mental disparada por aquela expressão que se refere ao objeto. O interpretante é a atividade mental desencadeada pela

[15]O trabalho do Peirce está em artigos escritos no final do século passado. Eles foram reunidos em *The Collected Papers of Charles Sanders Peirce* 1931-1938. Utilizamos a tradução brasileira *Semiótica* da Editora Perspectiva, coleção estudos 46, 2a. Edição, 1995.

[16] Na definição de Peirce o termo para este elemento é *representamen.* Em alguns trechos de seus artigos, Peirce utiliza o próprio termo *signo,* uma vez que este elemento é a parte perceptível do signo. Entretanto, optamos por utilizar o termo *expressão,* que faz parte da estrutura de signo na semiótica de Hjelmslev, mas com o sentido de *representamen* de Peirce.

expressão do signo - a realização de associações de idéias, ou a configuração de estruturas complexas de significados, e assim por diante.

De maneira abreviada, podemos exemplificar os elementos do signo de Peirce através da *expressão* <casa>, na língua portuguesa, que se refere ou representa um *objeto* (ou classe de objetos) estabelecido culturalmente e desencadeia na mente de todo aquele que é exposto a esta palavra uma cadeia indefinidamente longa e variada de associações de significados (por exemplo, <lar>, <família>, <proteção>, <aconchego>, e assim por diante) ou *interpretantes*. Como se pode ver, a cadeia associativa (que emerge no processo de semiose) é marcada por fatores subjetivos, muito embora possa ser motivada por elementos objetivos do contexto em que o ouvinte do signo se encontra.

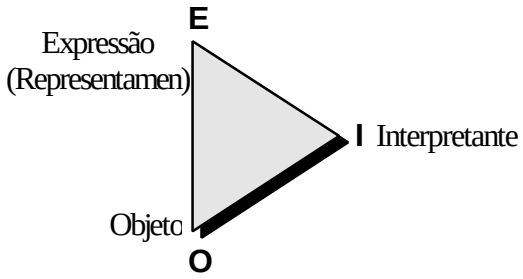


Figura 0-2: O Signo de Peirce

Vejamos agora o conceito de comunicação. Comunicação é, etimologicamente, o processo de por algo em comum - uma informação, conceito ou pensamento. Existem diversos modelos para o processo de comunicação, provenientes das mais diversas fundamentações teóricas. Ao invés de utilizarmos algum destes modelos, vamos descrever como tal processo deve ser visto a partir dos conceitos de Peirce. Utilizando o conceito de signo, podemos dizer que comunicação é o processo que ocorre entre duas ou mais pessoas quando uma delas produz algo (uma expressão) que conduz os interpretantes de todas as outras a referir-se a um objeto comum. Comunicação é, portanto, o que ocorre quando alguém produz algo que é, para si próprio, um signo (A) e é, também, para uma outra pessoa, um outro signo (B) cuja expressão e objeto convergem em (A). Os signos são distintos porque os interpretantes não são os mesmos, uma vez que eles são processos ilimitados que ocorrem nas mentes de cada um dos agentes. A expressão e o objeto são elementos que são postos em comum. Chamaremos o primeiro

agente de *produtor* - do signo (A) - enquanto que o segundo chamamos de *intérprete.* Utilizando termos mais convencionais, o produtor pode ser chamado de *emissor*, o intérprete de *destinatário*, e o signo de *mensagem.* Este modelo está ilustrado na Figura 0-3.

Para comunicar - colocar uma informação ou conceito em comum entre duas ou mais pessoas - faz-se necessário produzir uma expressão que conduz os *interpretantes* de todos os agentes envolvidos a referir-se a esta informação ou conceito. O maior problema que envolve comunicação é justamente saber qual expressão produzir. Os códigos são os sistemas que relacionam expressões a objetos. Estas relações quando compartilhadas entre os agentes podem produzir signos cujos objetos sejam o mesmo. Desta forma, os códigos oferecem regras que permitem aos agentes comunicar através da produção de expressões que possuam objetos associados. Voltaremos a discutir sobre códigos no capítulo 5.

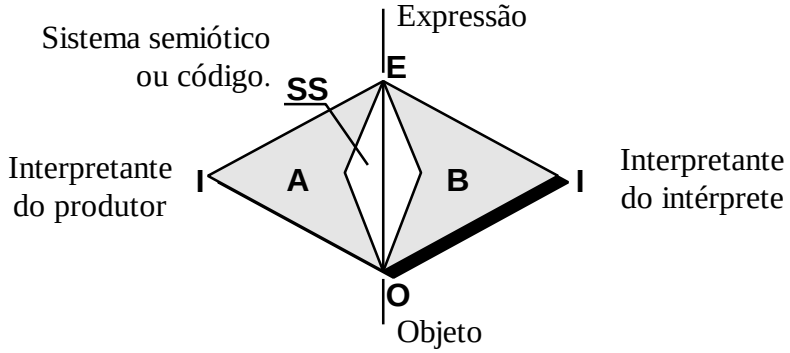


Figura 0-3: A comunicação na perspectiva semiótica.

Um Modelo para a Comunicação Designer-Usuário

Como na perspectiva central da Engenharia Semiótica o designer comunica-se com o usuário através da interface, ele deve projetar a interface de maneira a melhor transmitir o potencial de aplicação do sistema. Nesta seção vamos mostrar como o modelo de usabilidade pode ser expresso pela Mensagem do Designer, vista dentro de um paradigma conceitual de metacomunicação.

O modelo metacomunicativo possui quatro aspectos essenciais [Souza 96]:

- A interface de usuário veicula uma ***Mensagem do Designer***, realizando um ato comunicativo unidirecional, do designer para o usuário, e é portanto um ***Signo.***

- O significado da Mensagem do Designer (seu *objeto*) é o ***Modelo de Usabilidade.***
- O significado atribuído pelo usuário (seu *interpretante*) é uma abstração do Modelo de Usabilidade, e será referido como o ***Modelo conceitual de usabilidade do usuário*** (modelo adquirido de usabilidade).
- O significado pretendido pelo designer (seu *interpretante intencional*) é uma abstração do Modelo de Usabilidade, e será referido como o ***Modelo conceitual de usabilidade do designer*** (modelo potencial de usabilidade).

O significado pretendido pelo designer e o significado atribuído pelo usuário à mensagem da interface não são necessariamente idênticos, embora, obviamente, eles devam convergir e ser consistentes um com o outro.

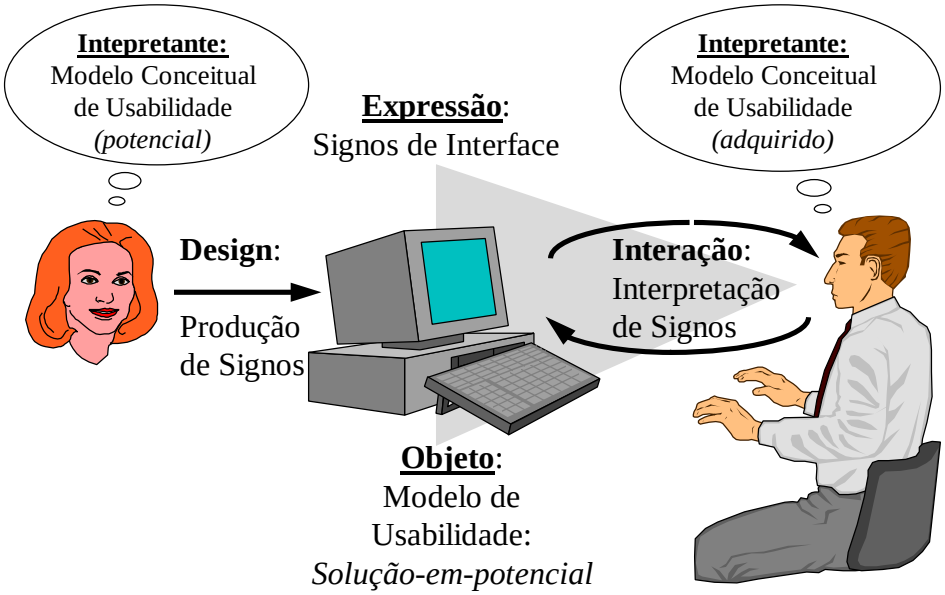


Figura 0-4: O modelo metacomunicativo da Engenharia Semiótica.

O designer pode comunicar o *modelo potencial de usabilidade* (modelo conceitual de usabilidade do designer) usando diferentes mídias tais como o manual do usuário, sistemas de ajuda e outros. Contudo, como a interface de usuário pode ser vista como uma *medium* e é nela que se passa a parte mais significativa do contato entre usuário e sistema, o designer deve utilizá-la como *medium* preferencial. O problema comum de todas estas *media* é que o designer normalmente não pode enviar a mensagem da

interface diretamente aos usuários. A mensagem do designer se concretiza como um desdobramento do processo de interação, não sendo possível que em um único turno interativo o usuário apreenda e compreenda a totalidade das noções e oportunidades que lhe são comunicadas. O processo de interpretação global é ativado e determinado pelas interações que ocorrem através do sistema. As mensagens usuário-sistema são as *mensagens da interação.* A partir da troca de signos através de entradas e saídas (E/S) pelo display, o usuário constrói uma cadeia de interpretantes [Peirce 31] que eventualmente se cristaliza no significado atribuído pelo usuário para a mensagem global - o *modelo adquirido de usabilidade* (modelo conceitual de usabilidade do usuário). O esquema proposto está ilustrado na Figura 0-4.

Exemplo: o GHS

Nesta seção vamos introduzir um outro exemplo com o objetivo de demonstrar como esta perspectiva teórica revela importantes aspectos do design. Analisaremos duas aplicações de software bastante simples que ilustrarão os conceitos abstratos discutidos ao longo desta tese. Mostraremos como a Engenharia Semiótica revela problemas de um dos protótipos - o GHS1 - e orienta o design do outro protótipo - o GHS2.

O GHS

O *Sistema de Ajuda Geral (General Help System - GHS)* é um protótipo de aplicação de software que visa ajudar o usuário a encontrar recursos na Internet [de Souza 92]. Através da Internet pode-se estabelecer a comunicação com pessoas, utilizar serviços de processamento remoto, ou transferir e enviar arquivos em sistemas computacionais que estejam conectados à rede, em qualquer lugar do mundo. Esta é uma taxonomia básica proposta para os recursos disponíveis em rede. O GHS é um sistema de informações trivial, dotado de uma base de dados relacional que permite ao usuário armazenar *anotações* sobre *pessoas, serviços e arquivos remotos.*

Nós desenvolvemos dois protótipos do GHS usando o Microsoft Visual Basic 3.0 para estes estudos de casos. Na figura e na figura , nós apresentamos as telas principais de cada um destes protótipos. Em ambos os protótipos a funcionalidade do programa é a mesma uma vez que o núcleo da aplicação é descrito pelo mesmo programa fonte. As interfaces oferecem diferentes modelos de interação. A usabilidade de cada aplicação

varia em cada caso. Assim, eles são dois signos diferentes quando utilizados pelos usuários.

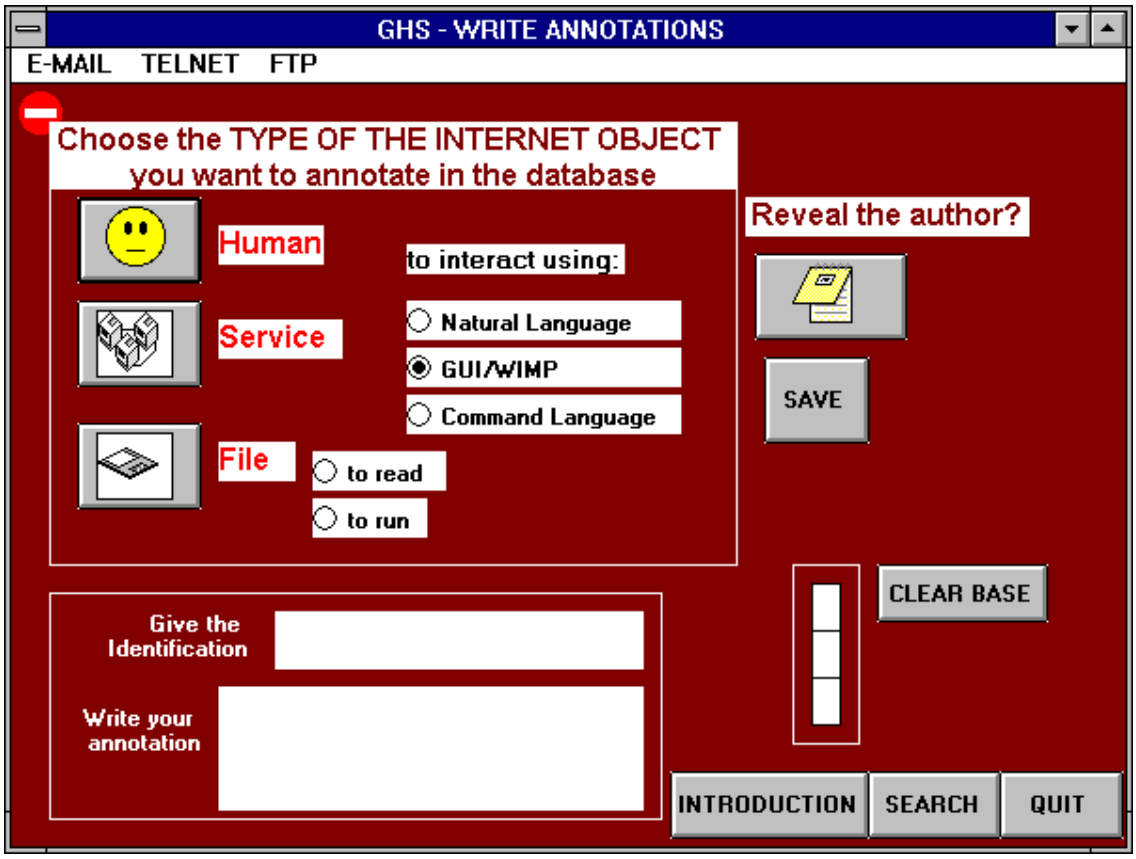


Figura 0-5: GHS1

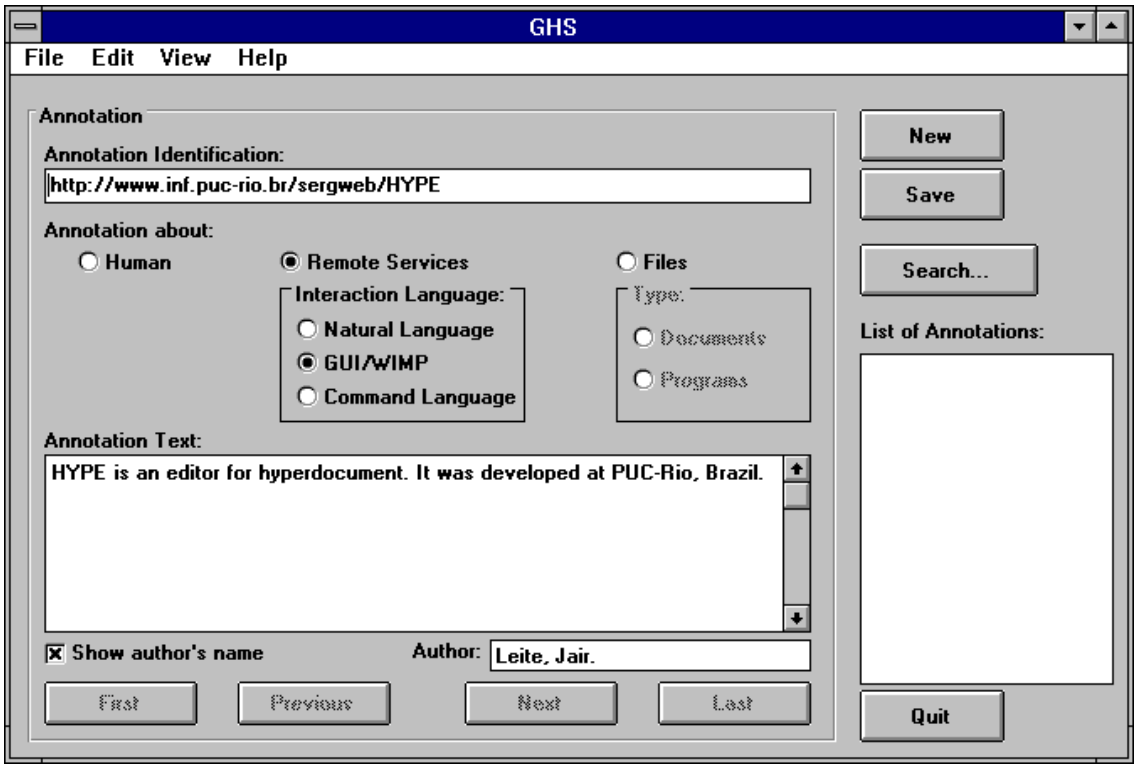


Figura 0-6: GHS2

Uma análise dos protótipos

O modelo de usabilidade de cada aplicação GHS é formalmente definido pelo programa interpretado em Visual Basic (VB). O núcleo da aplicação que define a sua funcionalidade é um programa de acesso a uma base de dados de anotação onde os usuários podem armazenar e procurar anotações sobre pessoas, serviços ou arquivos na Internet. A estrutura da anotação possui os campos de dados *anotação, identificação, nome-do-autor, tipo-do-recurso (pessoas, serviços ou arquivos)* e *texto* da anotação, todos definidos como um tipo de dados *string*. O usuário armazena uma anotação fornecendo valores para estes campos da base. As diferentes interfaces de usuário de cada aplicação definem diferentes modelos de interação. Assim, cada protótipo da aplicação implementa e define diferentes modelos de usabilidade. Eles serão referidos como modelo de usabilidade-GHS1 e modelo de usabilidade-GHS2.

Na perspectiva da Engenharia Semiótica o modelo de usabilidade é comunicado através da Mensagem do Designer. Com os dois protótipos podemos mostrar, mesmo através de instantâneos (estáticos) de tela, que diferentes mensagens são comunicadas.

Lembremos que analisando apenas a figura e a figura não é possível adquirir o modelo de usabilidade global de cada aplicação, somente acessível através do uso exaustivo da aplicação e das trocas de dados de E/S. Lendo dados, escrevendo texto, pressionando botões o usuário vai aprendendo sobre o que pode ser feito e como deverá proceder para atingir seus objetivos. Ou seja, vai adquirindo os interpretantes da aplicação. Os signos da interface não são apenas os textos e ícones que aparecem na tela da interface, mas incluem os dados e comandos produzidos pelo usuário, ou qualquer distinção que pode revelar algo sobre o modelo de usabilidade da aplicação. O designer deve estar consciente deste fato na elaboração da sua mensagem.

A utilização de cada uma das aplicações determina diferentes interpretantes para os seus usuários. Esses interpretantes, o modelo conceitual de usabilidade do usuário, refere-se àquilo que projetado pelo designer, que tem em mente um modelo conceitual de usabilidade em potencial. Para cada uma das interfaces apresentadas na figura e na figura , um mesmo usuário adquire conhecimentos distintos a respeito da aplicação. Os diversos motivos pelos quais isto se sucede podem ser facilmente esclarecidos pela perspectiva da Engenharia Semiótica. O principal deles é devido ao fato de que usuários e projetistas utilizam linguagens diferentes (ver seção 4.3). O projetista especifica e implementa o modelo de usabilidade a partir do conhecimento que ele tem do domínio de aplicação, das tarefas e dos usuários, utilizando-se de linguagens de especificação funcional e de programação. A linguagem de programação é descritiva e estática, enquanto que a linguagem da interface é dinâmica e interativa.

Um outro motivo é que o usuário pode utilizar a aplicação de uma maneira não prevista pelo designer. Neste caso, o modelo conceitual de usabilidade do usuário é diferente do modelo conceitual de usabilidade do designer porque o usuário descobriu uma nova funcionalidade para a aplicação. Por exemplo, como os sistemas GHS permitem o armazenamento e recuperação de informações o usuário pode utilizá-las como agenda para realizar anotações sobre números de telefones, ou como banco de dados para informações sobre filmes de sua videoteca particular. Neste caso, mudou a aplicação, muito embora o programa subjacente permaneça o mesmo.

As Mensagens de Interação e a Mensagem do Designer

Usuários que são familiarizados com o estilo de interface Microsoft Windows sabem que *botões de comando* são widgets normalmente utilizados para ativar funções, enquanto que o widget *botão de opção* se aplica à escolha exclusiva de uma-de-muitas opções. Na Engenharia Semiótica, esta utilização é referida como o significado do widget. No GHS1, botões de comandos são utilizados para se fazer escolhas a respeito do *tipo do recurso* Internet para o qual se está fazendo uma anotação. Neste caso o significado do *botão de comando* é diferente do significado usual para usuários do Microsoft Windows. Isto pode levar a interpretações erradas do modelo de usabilidade, como foi constatado num teste informal com usuários do protótipo.

No GHS2, widgets *botões de opção* foram utilizados para melhor comunicar ao usuário que ele deveria fazer uma escolha entre as três opções de tipos e, assim, determinar o valor de um atributo da anotação. Eles revelam ainda que a ação que o usuário deve fazer para colocar o valor no campo da base é apenas clicar no widget correspondente.

Um outro problema com a interface do GHS1 é que o usuário, em alguns casos, só tem como saber que um dos atributos foi escolhido se ele observar que os ícones em cada botão mudam, indicando a escolha da opção correspondente (por exemplo, o signo apresentado no botão correspondente a pessoas aparece sorrindo se esta for a opção escolhida). Como esta mudança não é muito perceptível, i.e., o usuário não conhece este código, fica difícil saber qual escolha foi feita. No GHS2, a utilização de widgets mais usuais e adequados (segundo [Microsoft 95]) oferece um *feedback* convencional para o usuário informando diretamente qual opção está ativa. Este feedback é o ponto preto no centro do widget.

Vamos considerar um outro aspecto da comunicação designer-usuário. É possível ver que a estrutura da anotação é melhor visualizada no GHS2 do que no GHS1, permitindo uma interpretação mais aproximada daquela concebida pelo designer. Isto ocorre porque no GHS2 são utilizados widgets com seus significados mais usuais e dispostos como objetivo de revelar importantes aspectos sobre seus componentes e seus relacionamentos.

Outro caso que exemplifica aspectos de interpretação da mensagem do designer está na barra de menus. Em geral, os aplicativos com interfaces estilo WIMP apresentam a seqüência *Arquivo, Editar, Exibir,... e Ajuda,* com uma estrutura em cascata (pull-down) contendo diversos comandos. No GHS1, a barra de menu apresenta comandos para ativar outras aplicações, o que é uma estrutura não usual. O conhecimento do uso convencional desencadeia interpretantes nos usuários que são diferentes daqueles que o projetista tinha em mente. No GHS2, a estrutura de barra de menus é utilizada para categorizar comandos de acordo com certos critérios tradicionalmente adotados na maioria das aplicações. *Arquivo* agrupa comandos associados a entidades externas, o *Editar* contém opções como *cortar, copiar* e *colar,* e *exibir* permite a visualização e busca das anotações.

É importante observar que, por um erro de tradução das versões brasileiras de aplicativos da Microsoft, torna-se difícil a interpretação deste modelo de interação por menus. Através do menus *File, Edit, View,...* o designer queria comunicar categorias de comando relacionados a *arquivar, editar e exibir.* A tradução de File por Arquivo deturpa bastante a intenção do designer. Por outro lado, a presença dos signos EMAIL, TELNET e FTP no GHS 1 revela uma vantagem. Eles possibilitam ao usuário uma interpretação mais situada da aplicação da funcionalidade no seu contexto.

Embora tenhamos afirmado que é interessante utilizar widgets e estruturas usuais isso não deve ser visto com uma regra categórica de design. Nosso argumento é que usuários realizam um trabalho de interpretação a partir dos signos de interfaces que foram escolhidos pelo designer. Os widgets são parte de uma mensagem global e devem comunicar o modelo de usabilidade. A decisão a respeito de quais devem ser os widgets, sejam eles convencionais ou não, adequados ou não, é uma decisão de design importante para a qual a Engenharia Semiótica deve apresentar subsídios teóricos.

Os usos ou significados tradicionalmente adotados podem ser vistos como a aplicação de um código incipiente que visam facilitar a interpretação da funcionalidade e do modelo de interação. Os widgets do VB utilizam o estilo de padrão do MS Windows que determina significados preferenciais para cada um deles. A escolha de widgets

convencionais[17] e a sua disposição, segundo regras de *layout* também já estabelecidas, são passos importantes na tentativa de estabelecer uma linguagem para a comunicação designer-usuário.

O GHS1 foi projetado sem que o designer tivesse em mente os conceitos da Engenharia Semiótica a respeito de mensagem e significado. No GHS2 o projetista utilizou códigos informais para expressar o modelo de usabilidade. Entretanto, ao mesmo tempo que o GHS2 exemplifica vantagens, ele revela as fraquezas do código utilizado. Estes aspectos serão discutidos em detalhes nos capítulos 5 e 6.

A Aplicação de Software como Signo

De acordo com a teoria semiótica que estamos considerando, para que a comunicação designer-usuário seja possível é preciso considerar que aplicações de software são signos, formados por signos, e que geram e interpretam signos. Esta perspectiva estende a proposta original de [de Souza 93] e apresenta uma justificativa mais poderosa para a Engenharia Semiótica, pois revela que a engenharia de software e as atividades de programação e de interação proporcionam diferentes interpretações da usabilidade de um software. Estas considerações justificam a necessidade de se conceber o sistema computacional através de uma abordagem semiótica baseada na produção de signos.

Sistemas e Signos

Sistemas computacionais são dispositivos eletrônicos que adquirem funcionalidade e usabilidade pela sucessiva interpretação de padrões simbólicos produzidos na interação como o usuário e processados pelo sistema. Estes padrões são programados através de uma linguagem de programação. O software não é interpretado através da observação direta da "*substância*" eletrônica (usando um termo da semiótica de Hjelmslev), mas por camadas de interfaces que convertem ou (de)codificam, algoritmicamente, sinais em expressões inteligíveis. Teclados numéricos, tabelas ASCII, widgets, compiladores e linguagens de programação, *debuggers*, dentre outros, são as interfaces que tornam os signos mais facilmente interpretados. As interfaces transformam os sinais eletrônicos binários em expressões que podem ser interpretadas por usuários e programadores. A interface de usuário é um subconjunto destas interfaces. Programar e

[17]Convenção estabelecida pelo padrão Windows.

usar computadores são atividades de criação de significados através da intensiva produção de signos. As interfaces de usuários e as linguagens de programação permitem que usuários e programadores lidem com representações que lhes são mais familiares.

Voltando à questão da subtração, como exemplo, vamos imaginar o processo de subtrair dois números inteiros. Podemos escrever este processo num pedaço de papel, usando uma notação aritmética, de maneira que podemos armazená-lo, lê-lo ou comunicá-lo a alguém. O que está num pedaço de papel são apenas marcas de tinta, mas que estão codificadas de maneira que alguém que conheça o código aritmético pode interpretá-las, compreendê-las e aplicar seu princípio a diversos números inteiros. A descrição do processo no papel torna-se um signo quando lido e interpretado por alguém. Este processo, de acordo com o conceito de *semiose ilimitada* de Peirce, gera uma cadeia de possíveis aplicações do processo de subtração. Estes processos mentais de compreensão do algoritmo que geram as suas aplicações são *os interpretantes*. O *objeto* deste signo compreende todas as aplicações do algoritmo para a subtração. Sua *expressão* são as marcas de tinta num pedaço de papel (o medium) de acordo com a notação aritmética (o código). Este algoritmo pode ser codificado em diversos outros códigos, como numa notação gráfica ou numa linguagem de programação.

Suponha agora que queiramos fazer uma subtração usando um programa de computador. É necessário codificar o algoritmo numa linguagem de programação indicando como realizar os cálculos necessários, receber os dados de entrada e enviar os resultados para o usuário. Este programa quando executado por um computador pode ser aplicado na resolução de inúmeros problemas dos seus usuários. A aplicação do software na resolução de problemas é o objetivo tanto de quem concebeu e programou o sistema quanto dos próprios usuários. Entretanto, esta aplicação apenas ocorre quando o usuário tiver conhecimento sobre o modelo de interação e de funcionalidade do software (sabendo o que e o como fazer). Estes modelos são abstratos e apenas podem ser contruídos, comunicados e interpretados com o auxílio de sistemas de signos, o que pode ocorrer através de manuais, sistemas de ajuda ou da interface de usuário. Quando este processo ocorre através da interface de usuário durante o processo de utilização, podemos dizer que ela é a expressão de um signo cujo objeto é o modelo de usabilidade necessário para a aplicação do sistema

A interface de usuário é a expressão que gera no usuário um interpretante. O interpretante é a cadeia de intepretações gerada a partir dos signos de interação através da interface que se refere ao objeto. O objeto é a aplicação do sistema na resolução de problemas do domínio. Dizemos, portanto, que quando sistema, através da sua interface de usuário, gera no usuário um processo mental que o capacita a aplicá-lo na resolução de problemas ele é um *signo*. Esta perspectiva nos permite observar ***aplicações de software como signos***. Este esquema está apresentado na Figura 0-7.

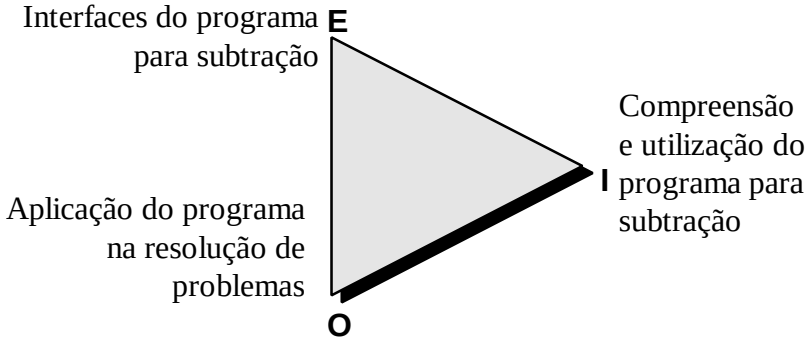


Figura 0-7: Aplicação Subtração com Signo

Para um designer, a expressão que descreve estes modelos de interação e de funcionalidade pode ser o modelo de especificação. Um programador pode utilizar a linguagem de programação para codificar e interpretar os modelos funcionais e de interação. Programas fonte, modelos de especificação e interfaces de usuário são, portanto, expressões diferentes que devem ser construídas para um mesmo objeto: o modelo de usabilidade que permite a aplicação do software na resolução de problemas. Em todos estes casos, temos um processo de signo ativado pelas diferentes interfaces de um sistema. Genericamente podemos considerar que aplicações de software são signos para todos os que observam as suas interfaces.

De acordo com a teoria semiótica as interfaces do sistema com programadores e usuários são as diferentes *expressões* que se podem produzir no *medium* computacional. O *objeto* do signo é o conjunto de todas as interpretações e possibilidades de aplicação do sistema que podem ser obtidas durante a interação com o sistema. Percebe-se que na perspectiva semiótica este conjunto é aberto (não enumerável), não sendo possível

capturar todas as interpretações possíveis de um signo, uma vez que este processo é ilimitado (veja-se o conceito de semiose ilimitada). Este ponto é teoricamente interessante por traçar nitidamente a fronteira de computabilidade que distingue semântica de semiótica. Semântica é computável, enquanto que semiose não é um processo algorítmico.

Tal como vários outros pesquisadores já haviam enfatizado (e.g. [Andersen 90, Nadin 88, Nake 94], a semiótica apresenta uma perspectiva na qual aplicações de software encontram uma base teórica adequada para serem estudadas. Este exemplo vem ao encontro das argumentações destes autores que serão reforçadas daqui por diante pela Engenharia Semiótica.

A perspectiva de aplicações de software como signo vai ao encontro do conceito de software como *virtualidade,* colocado em [Winograd 96] como fundamental para o design de software. Uma aplicação de software é uma máquina abstrata ou virtual que emerge destes padrões de sinais processados pelo computador. Esta *virtualidade* é um *signo* cujo significado é interpretado pelos usuários como sendo a sua *usabilidade*.

Códigos e Interpretantes da Aplicação

A abordagem da Engenharia Semiótica nos ajuda a entender alguns problemas não solucionados pela Engenharia de Software no desenvolvimento de sistemas. A perspectiva da aplicação como signo revela o importante papel que as linguagens possuem na interpretação do modelo de usabilidade.

Designer, programadores, e usuários do sistema utilizam diferentes sistemas semióticos para produzir e/ou interpretar as diferentes interfaces para o modelo de usabilidade. Além disto, cada uma delas são expressões parciais deste signo. Para o usuário a expressão é formada pelas diferentes mensagens de interação veiculadas pelos objetos de interface. Para o programador o signo é determinado pela expressão *programa fonte* que deve ser interpretada pelo código *linguagem de programação.* Um programa fonte é a expressão do signo cujo objeto é o conjunto de todos os caminhos possíveis de execução. Este objeto é definido formalmente pela semântica da linguagem de programação. O programa fonte gera como interpretantes mais imediatos o funcionamento do programa. Apenas programadores que conheçam a fundo a semântica

da linguagem e consigam realizar abstrações a partir de algoritmos e estruturas de dados codificadas conseguirão chegar a interpretantes de usabilidade. Outras notações (expressões) da engenharia de software como diagramas da arquitetura do software, algoritmos descritos em linguagem natural estruturada, modelos gráficos de estruturas de dados, linguagens de especificação funcional, dentre outras, visam descrever abstratamente o funcionamento e não a usabilidade.

Por utilizarem expressões diferentes, os interpretantes do designer, do programador e do usuário não são idênticos ao modelo semântico determinado pela linguagem de programação. Os métodos de concepção de software não devem utilizar-se apenas de modelos de abstratos de funcionamento, mas de modelos que permitam antecipar a usabilidade. As técnicas de prototipação e de design *baseado-em-cenários*, estão de acordo com esta perspectiva da Engenharia Semiótica. Além disso, o designer da interface deve se utilizar de linguagens que envolvam as mensagens de interação.

As figuras a seguir representam os diferentes signos para o exemplo do programa subtração que ocorrem quando o intérprete é o programador (Figura 0-8a) e quando é o usuário (Figura 0-8b). Para o programador, a linguagem de programação restringe os interpretantes em torno dos conceitos de *transformação de estados de dados e espaços de memória*, dificultanto a interpretação do aplicação do software. Para o usuário, dependendo de como a interface é concebida, ele pode interpretar um bom ou um ruim modelo de usabilidade. Uma possível solução para este problema é objeto dos próximo capítulos.

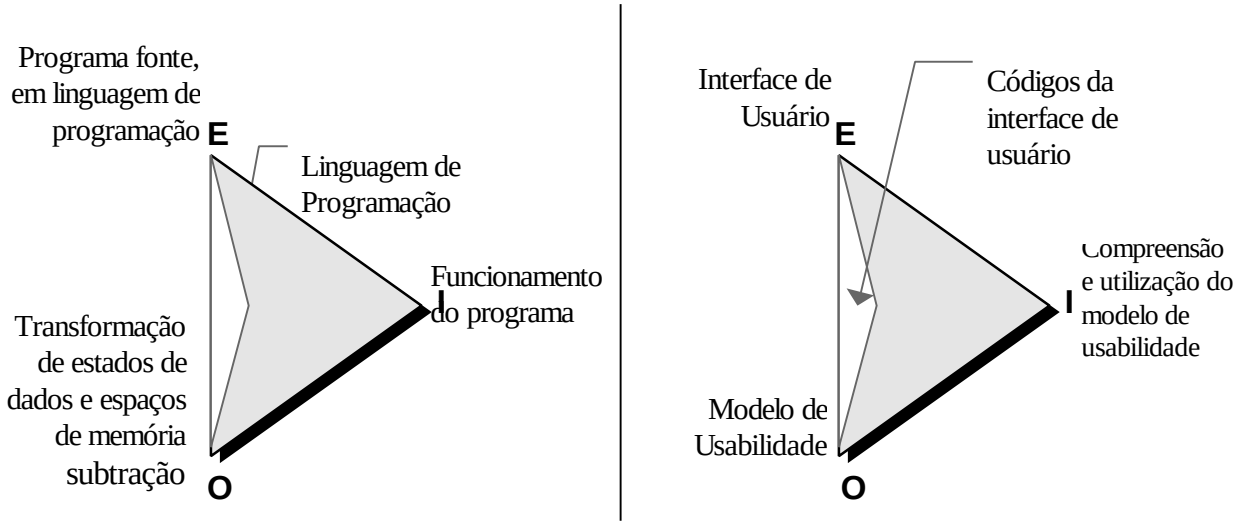


Figura 0-8: Signos da Aplicação Subtração: (a) interpretado pelo programador e (b) interpretado pelo usuário

O design para usabilidade como produção de signo

A discussão acima revela a importância da interface para que o usuário adquira o modelo de usabilidade de um sistema. Por este motivo, o design da aplicação requer a produção de interfaces que comuniquem o modelo de usabilidade. O requisito fundamental que a perspectiva da Engenharia Semiótica coloca para o desafio de usabilidade é que o design é uma atividade de produção de signo.

O que a Engenharia Semiótica diz é que no design do modelo de interação deve-se estar realizando uma comunicação. Por exemplo um botão serve ao acionamento (interação) ao mesmo tempo que a sua imagem (*affordance*) comunica qual é a ação que ele deve fazer (metacomunicação). Ele é expressão e conteúdo. O objetivo a ser alcançado por este trabalho é oferecer instrumentação para a metacomunicação que é necessária ao design.

O DESIGN DE INTERFACES DE USUÁRIO COMO PRODUÇÃO DE SIGNO

No enfoque conceitual da interação entre usuário e sistemas, modelos da interface e do processo de interação são dependentes da abordagem teórica utilizada. O capítulo anterior apresentou a perspectiva geral de que aplicações de software são signos. De acordo com a definição de Peirce, vista no capítulo anterior, um signo é um processo no qual algo (a expressão) está no lugar de outra coisa (o objeto) para alguém, através da ativação de interpretantes na sua mente. Vimos exemplos (o GHS e o programa de subtração) de como sistemas interativos podem ser vistos como signos e é através de suas interfaces (suas expressões) que os interpretantes são ativados.

A abordagem da Engenharia Semiótica precisa de um modelo para a *aplicação de software como signo* e para o seu *design como um processo de produção de signos* que possa ser utilizado na elaboração de métodos e ferramentas de apoio à construção da Mensagem do Designer. Neste capítulo mostramos como processos de produção de signos da teoria semiótica podem ser adaptados para o design de interfaces de usuário e apresentamos modelos para a *expressão* e o *objeto* da *Mensagem do Designer*. Veremos também que é preciso desenvolver *Sistemas Semióticos* compostos por *tipos-signos* que possam ser empregados no apoio ao processo de design.

Fundamentos para o Design como Produção de Signo

Projetar interfaces é, portanto, projetar uma mensagem complexa, interativa, unidirecional, destinada a usuários de aplicações computacionais. Vimos no exemplo do GHS que diversos elementos da interface podem possuir significados distintos para o designer e para o usuário. Botões, palavras, cores, menus, etc., quase tudo na interface

tem o potencial de ser signo. O designer necessita controlar este processo de comunicação para melhorar a usabilidade do sistema. Ele precisa projetar a interface consciente de que está projetando um signo cuja expressão é formada por outros signos que devem ativar interpretantes que conduzam ao Modelo de Usabilidade.

Os modelos descritivos e instrutivos do processo de design de interfaces até o presente momento raramente explicitam este fenômeno de comunicação. Exceção pode ser feita aos trabalhos de [Nadin 88; Kammersgaard 88; Andersen 90]. Mas a maioria enfatiza problemas cognitivos como [Card, Moran & Newell 83; Norman 86] certamente derivados de fenômenos de comunicação, na tentativa de clarear a *meta* da comunicação, sem contudo elucidar o *meio* de efetivá-la. Esta segunda etapa, ausente nos modelos cognitivos, é o que a Engenharia Semiótica pretende atacar.

A exemplo do que se passa nas abordagens de inspiração cognitivista (revisadas anteriormente), a Engenharia Semiótica deverá, para tornar-se operacional no cenário de design de interfaces, prover não apenas explicações e previsões sobre fenômenos de comunicação designer-usuário, mas também, e sobretudo, subsídios metodológicos e ferramentais que permitam ao designer atuar de maneira mais objetiva e bem sucedida na direção de uma boa comunicação com os usuários das interfaces por ele produzidas. Nas seções seguintes vamos apresentar alguns conceitos de semiótica do processo de produção de signos que fornecem a base teórica para atingirmos tais objetivos.

Medium

A expressão de um signo é consituída por distinções que podem ser percebidas por alguém. Um *medium* pode ser qualquer coisa através da qual se pode produzir distinções que podem ser percebidas pelos canais sensoriais de uma pessoa. Eco considera o medium como um *contínuo expressivo* que deve ser estruturado para a produção de expressões [Eco 76].

O ar, a luz e suas refrações e reflexos são os exemplos de *media* mais comuns. Estímulos táteis, odores e sabores também são *media*. É também considerado *medium* qualquer dispositivo artificial que possa ser utilizado no armazenamento e produção de distinções que podem ser codificadas ou decodificadas em distinções perceptíveis pelos canais sensorais. A fala e a música são percebidas pelo canal auditivo através do *medium*

ar. Escrita, pintura e fotografias são percebidas pelo canal visual através do *medium* "luz refletida através de papel". Uma música pode ser armazenada no medium artificial *compact disc* que é ótico e eletrônico ou numa *fita magnética* que é magnético e eletrônico. Para chegar ao canal auditivo a música precisa ser decodificada em distinções sonoras produzidas por alto falantes.

Os objetos da interfaces, como veremos mais adiante, funcionam como medium através do qual signos de interface podem ser construídos. Foge ao escopo do nosso trabalho considerar os diversos tipos de *media* e as diversas tecnologias a elas associadas. Vamos nos restringir às tecnologias mais comuns de interface GUI/WIMP que podem funcionar como *contínuo expressivo* através do qual o designer pode construir mensagens.

Tipos-Signos

Num processo comunicativo, quando se tem a intenção de produzir um determinado signo, pode-se utilizar tipos de signos preexistentes que, quando conhecidos tanto pelo produtor quanto pelo intérprete, restringem o processo de interpretação "em torno" do objeto. Estes tipos de signos são padrões abstratos de expressões para os quais o conhecimento prévio e a cultura da sociedade associou uma outra entidade, o seu objeto, atribuindo-lhes um significado. Os ***tipos-signos*** têm por objetivo orientar a produção e interpretação de signos de maneira que o interpretante do consumidor seja o mais próximo possível do pretendido pelo produtor.

Um *tipo-signo* refere-se ao conhecimento necessário que possibilita a ocorrência do signo. Eco considera que um *tipo-signo* define regras que associam ***tipos*** (propriedades) ***expressivos*** a ***tipos*** (propriedades) ***semânticos*** [Eco 76]. Estas regras, quando conhecidas, conduzem as expressões aos interpretantes que se referem ao objeto do signo. O tipo expressivo é um modelo pré-definido a partir do qual se pode elaborar um *token* expressivo, um instância do tipo, que é a expressão de um signo. Quando este modelo é definido a partir de um repertório de tipos básicos (vocabulário) e de regras para a sua combinação (gramática), temos um *sistema expressivo* ou *sintático.*

O tipo semântico refere-se a uma unidade de conhecimento que é associada ao tipo expressivo como sendo o seu significado preferencial. É esta associação que permite

que um *token* expressivo seja tomado, por alguém, como ocupante do lugar de outro: o *token* semântico. São as correlações entre os tipos expressivos e semânticos que oferecem as condições cognitivas para que algo funcione como um signo para alguém. Isto ocorre, portanto, quando uma expressão (*token* expressivo) é instância de um tipo expressivo que, de acordo com o conhecimento de alguém, está relacionado com tipos semânticos para os quais pode existir um *token* semântico determinado pelos processos e relações cognitivas.

Uma palavra é um tipo que podemos instanciar quando quisermos comunicar algo sobre os objetos do mundo ou conceitos que os falantes da língua associaram a ela. A palavra *cadeira* por exemplo, pode ser utilizada para significar um objeto específico, uma classe de objetos, a idéia de sentar e vários outros conceitos. Isto só é possível para quem conhece a regra que estabelece esta associação. As sentenças de uma linguagem constituem tipos-signos complexos que estabelecem associações entre as regras gramaticais, que permitem articular as palavras, e os seus significados (os tipos semânticos).

Os símbolos das placas de trânsito são tipos-signos estabelecidos pelas autoridades competentes cujas regras que associam a expressão ao objeto todos os motoristas devem aprender. A placa *pare*, por exemplo, deve ter cores e formas que obedeçam a um tipo estabelecido. As figuras utilizadas para informação em lugares públicos (tais como aeroportos) são tipos que utilizam regras que devem ser conhecidas por um grande número de pessoas de maneira a evitar conhecimentos mais específicos de uma determinada cultura.

Uma nuvem no horizonte pode ser expressão do objeto *vai chover* para aqueles que sabem que *chuva é proveniente de nuvem*. Para um especialista, os diversos tipos de nuvens constituem tipos expressivos distintos para diversos fenômenos metereológicos distintos. Por exemplo, temporal, garoa, neblina e frio são exemplos de tipos semânticos que um metereologista associa, em seu sistema de tipos-signos, aos diversos tipos de nuvens.

No escopo de interfaces de usuário os *widgets* de um certo padrão de interface são os exemplos mais próximos de tipos-signos. Um padrão de interface visa especificar a

aparência e o comportamento de cada widget (sua expressão) e quando eles devem ser utilizados (seu significado). Um botão numa interface quando produzido seguindo um padrão de *widgets* é instância de um tipo expressivo que pode ter como significado *aperte para acionar uma função*. Os ícones de pastas e documentos utilizados nas maioria das interfaces *desktop* são tipos expressivos cujos tipos semânticos associados são os conceitos computacionais de *diretórios* e de *arquivos.*

A maioria dos padrões de interface e das ferramentas de design que os seguem não foram desenvolvidos com o objetivo de ser um sistema de tipos-signos de interface que apóiem a produção da Mensagem do Designer. Eles não indicam ao designer como os elementos do Modelo de Usabilidade que ele concebe devem ser correlacionados com os diversos tipos de *widgets* que as ferramentas oferecem.

Sistemas Semióticos

Ao longo do tempo, os tipos-signos podem ser estruturados e articulados e tornarem-se sistemas de tipos-signos ou ***sistemas semióticos***. *Sistemas semióticos* são estabelecidos entre dois ou mais agentes de uma cultura ou domínio específico e devem ser utilizados na produção de expressões interpretáveis como signos. Um *sistema semiótico* estabelece tipos de signos gerais produzindo as regras que gerarão as ocorrências concretas - a produção do signo [Eco 76, pag. 42].

Um sistema semiótico tem como estrutura básica (1) um sistema de tipos expressivos para a produção de *token*s expressivos, denominado de ***sistema expressivo* ou *sintático***, (2) um sistema de tipos semânticos, denominado de ***sistema semântico,*** e (3) ***regras de correlação*** entre os elementos de cada um dos sistemas.

Um sistema semiótico estabelece regras de conhecimento que correlacionam um elemento do sistema expressivo a um elemento do sistema semântico. No processo de produção deve-se instanciar uma expressão a partir de um tipo do sistema expressivo que esteja correlacionado a um tipo do sistema semântico. A interpretação se dá pela aplicação das regras do sistema semiótico que conduzam a expressão ao objeto através dos seus tipos.

Códigos, hipocódigos e hipercódigos

A teoria dos códigos de Eco considera que sistemas semióticos podem ser classificados em *códigos, hipocódigos e hipercódigos* [Eco 76]. Um *código* é um sistema semiótico com um sistema sintático e um sistema semântico bem definidos. O sistema sintático é composto por um vocabulário e por regras gramaticais que permitem articulá-lo. O sistema semântico é composto por unidades e categorias conceituais relacionadas entre si. Existem códigos elementares onde a sintaxe é composta apenas por um vocabulário de tipos-signos que permite a produção direta de expressões não articuladas. Outros podem ser definidos por gramáticas e semânticas formais. As linguagens de programação são exemplos claros de códigos com sintaxe e semântica articuladas definidas de maneira formal. Já as linguagens naturais são exemplos de códigos onde a sintaxe e a semântica são bastante complexas, mas definidas de maneira não-formal e pragmática.

Um hipocódigo é aquele que ainda não está totalmente definido e estabelecido. Este sistema é utilizado de maneira informal num processo comunicativo e seus sistemas semântico e sintático não possuem unidades claras e bem especificadas. Regras de diagramação e uso de cores em design gráfico, quando explorados pelos editores de um jornal ou revista para comunicar algo, são exemplos de um hipocódigo. "A hipocodificação pode definir-se como a operação pela qual, na ausência de regras mais precisas, porções macroscópicas de certos textos são provisoriamente admitidas como unidades pertinentes de um código em formação, capazes de veicular porções vagas mas efetivas de conteúdo..."([Eco 76] p.123)

Um *hipercódigo* é um sistema cujos tipos-signos são composições de outros tipos ou são definidos em cima de um código base. Ele ocorre quando um código vem a ser um contínuo expressivo para a definição de novos códigos. Este novo código normalmente é um hipocódigo. Por exemplo, na comunicação escrita, a linguagem natural é um código base que pode ser utilizado com *contínuo expressivo* para a definição de um código gráfico. Este código gráfico, utilizado na comunicação visual, tem como unidades expressivas a variação de fontes, o tamanho das letras, a estrutura do texto e dos

parágrafos, etc. “Todas as regras retóricas e estilísticas que operam em cada língua constituem exemplo de hipercodificação”. [Eco 76].

Texto e extra-codificação

Na produção de um signo é possível utilizar-se de tipos-signos de diversos códigos. A um signo formado assim dá-se o nome de ***texto***. Um texto, segundo [Eco 76], é uma mensagem cuja estrutura é o resultado da coexistência de vários códigos. Ele é um retículo de diversos signos dependentes de diversos códigos, hipocódigos e hipercódigos num processo denominado ***extra-codificação***.

Um exemplo comum de *texto* é um jornal. Os signos do jornal são provenientes de sistemas semióticos diversos tais como a linguagem natural, a diagramação gráfica, a fotografia, o desenho e de recursos estilísticos e retóricos. A linguagem natural é o código usado na construção das sentenças das reportagens. A diagramação gráfica é um hipocódigo quando utilizada, por exemplo, para comunicar ao leitor quais são os pontos chaves de uma reportagem e quais são as informações auxiliares. Desenho e fotografia são produzidos a partir de tipos-signos isolados que não fazem parte de um sistema semiótico. A chamada *linguagem fotográfica* utilizada por especialistas poderia ser considerada um hipocódigo. Os recursos estilísticos e retóricos são um hipercódigo construído em cima da linguagem natural que pode ser utilizado por jornalistas para comunicar posições pessoais ou ironias a respeito dos fatos reportados.

As mensagens dos sistemas *multimedia* são construídas utilizando-se sistemas semióticos cujos tipos expressivos são de diferentes *media*. Um filme é uma mensagem *multimedia* que se utiliza da linguagem natural no *medium* auditivo e de hipocódigos cinematográficos no *medium* visual.

O design de software como atividade de extra-codificação

A Mensagem do Designer é um *texto* composto por uma diversidade de signos: signos do designer, do sistema e do usuário, produzidos intencionalmente ou não, e em tempo de design ou de interação. Estes signos, por sua vez, são produto da articulação de diversos códigos, hipocódigos e hipercódigos. Pode-se utilizar, por exemplo, palavras e frases da linguagem do usuário, códigos gráficos, cores, relações visuais e vários outros através dos quais se pode extrair os significados. A Mensagem do Designer como um

texto deve ser produzida por um processo de extra-codificação, ou seja, ela não obedece a um único código, mas a um repertório de tipos-signos provenientes de diversos códigos.

A estrutura de menus *pull-down*, presente na maioria das interfaces GUI/WIMP, exemplifica a característica de extra-codificação do *texto* interface (ver Figura 0-1). O menu *pull-down* é composto por uma *barra* onde são dispostos os cabeçalhos de cada menu. Cada cabeçalho deve indicar uma categoria de elementos que serão dispostos na cascata de sub-menus. Diversos signos provenientes de vários códigos subjacentes são utilizados nesta estrutura. O código lingüístico do usuário é utilizado para descrever as categorias e os elementos que são estruturados no menu. Na barra horizontal com os cabeçalhos é usado um hipocódigo visual que pode ser utilizado para comunicar ao usuário que ao clicar sobre uma palavra um menu pull-down será mostrado. Se, por exemplo, ao invés de mostrar um menu algo diferente acontecesse o código estaria sendo quebrado e poderia causar confusão nos usuários. Um terceiro código é utilizado no menu pull-down. Os símbolos "> " e "..." associados às palavras do menu constituem um código elementar que pode ser utilizado para comunicar se um outro menu pull-down será mostrado, ou se uma caixa de diálogo, ou ainda se é um comando pronto para ser executado. Temos ainda o código de cores preta e cinza que pode ser utilizado para comunicar se o elemento do menu está disponível para ativação ou não.

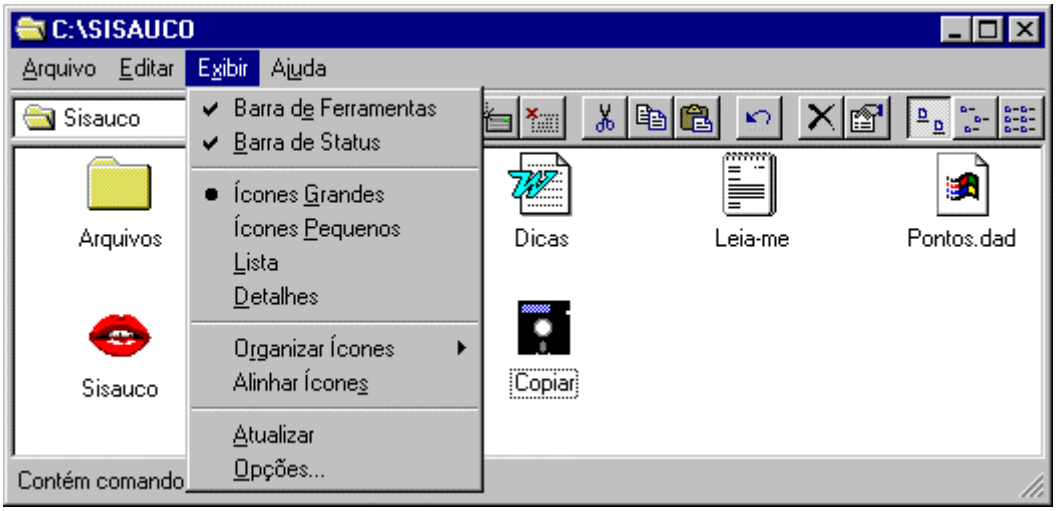


Figura 0-1: Exemplo dos códigos utilizados na interface.

Requisitos para o design como produção de signo

A *Teoria da Produção de Signo* de Eco permite caracterizar as diversas atividades envolvidas no design de interfaces em um esquema de atividades de produção de signos [Eco 76]. Adaptando-as para o design de sistemas interativos, podemos citar:

Segmentação do contínuo expressivo, ou "lexicalização", que permita identificação dos possíveis tipos expressivos. São exemplos de tipos expressivos os *widgets* que formam uma interface gráfica WIMP.

Segmentação do contínuo do objeto, ou modelagem, que permita a elaboração de meta-modelos da usabilidade. Por exemplo, identificar os componentes do Modelo de Usabilidade - dados, relações entre dados, funções que operam sobre estes dados e comandos que permitem o acionamento das funções. A engenharia de software apresenta diversos modelo da funcionalidade de um software, como por exemplo, *modelos de objetos, modelos de processo e fluxos de dados, modelos de estados*, e vários outros.

Estabelecimento de níveis de articulação, ou "sintatização", que são regras que permitam criar signos mais complexos a partir de tipos mais básicos. Por exemplo, botões de comandos podem ser articulados para compor um signo *barra de comandos. Widgets* de interfaces podem ser articulados em configurações espaciais - a maneira como eles estão dispostos na interface (*layout*) - e temporais - a ordem em que eles devem ser apresentados ao usuário.

Correlação entre expressão e objeto, ou "semantização". Os tipos-signos que podem ser utilizados pelo designer da interface devem ter tipos expressivos associados a tipos semânticos. Os tipos semânticos são as abstrações de unidades do Modelo de Usabilidade que deverá ser especificado pelo designer. Queremos identificar, por exemplo, que os *widgets botões de opção* têm como significado preferencial uma escolha entre valores pré-definidos de um tipo de dados. Estamos realizando assim uma correlação entre os elementos do sistema semântico (tipos de dados) e do sistema expressivo (*widgets*).

A partir deste quadro vemos que a elaboração de um sistema semiótico de apoio à produção da Mensagem do Designer requer a identificação de potenciais expressões (tipos expressivos) que possam ser construídas na interface e que estejam associadas com potenciais objetos ou significados para os usuários (elementos do Modelo de Usabilidade).

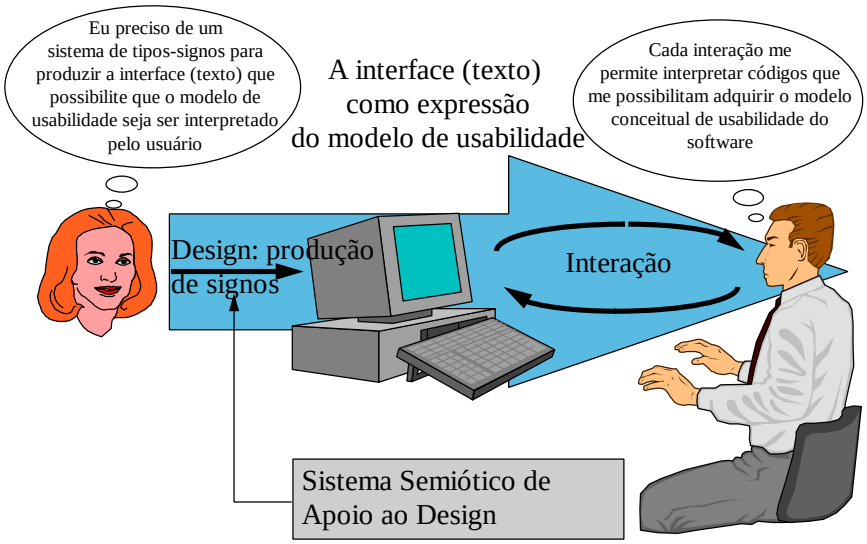


Figura 0-2: O Sistema Semiótico como ferramenta da Engenharia Semiótica.

Com base nestes e em todos os conceitos teóricos da teoria de Eco vistos nas subseções anteriores, concluimos que um sistema semiótico é um instrumento fundamental para o design de interfaces de usuário. A Figura 0-2 ilustra como a Engenharia Semiótica aborda o design de interfaces de usuário.

Os nossos próximos passos visam apresentar um modelo para as interfaces que permita identificar os seus tipos expressivos e um Modelo de Usabilidade cujos elementos possam associados aos primeiros, caracterizando-se como um sistema semiótico.

O Modelo de Usabilidade como Objeto da Mensagem do Designer

O Modelo de Usabilidade apresenta *soluções-em-potencial* que devem ser aplicadas aos problemas ou tarefas de um domínio (*o domínio de aplicação*). A concepção e especificação do Modelo de Usabilidade determina o conhecimento que o usuário deve ter e que precisa ser comunicado pelo designer através da interface. Ele deve determinar os componentes que o sistema possui e que permitem ao usuário utilizá-lo.

De acordo com os requisitos vistos na seção 5.1, é necessário fazer a "*segmentação do contínuo do objeto*" da mensagem para que possamos identificar as unidades de significados que, normalmente, um designer deve comunicar ao usuários. Entretanto, na literatura não identificamos modelos descritivos (meta-modelos) do

Modelo de Usabilidade que satisfaça ao propósito comunicativo. São propostas apenas notações e linguagens de especificação com as quais modelos específicos são elaborados. Por isso, é necessário apresentarmos um meta-modelo para o Modelo de Usabilidade de maneira a descrever as principais distinções conceituais.

Nas abordagens semióticas, o desenvolvimento de software é uma atividade de criação de significados através da produção de signos. A análise, o design, a programação e a avaliação dependem todos de processos de produção de interpretação de signos. Abordagens semióticas para análise de requisitos foram propostas por [Leite & Franco 93] e [Pimenta & Faust 97]. Programação é discutida em [Andersen 93] e o design nas várias abordagens discutidas no capítulo 3. A Engenharia Semiótica, apresenta uma descrição dos componentes de usabilidade a partir de uma perspectiva semiótica como objetivo de complementar as especificações funcionais e da interface já desenvolvidas pelas áreas de engenharia de software e de IHC.

Não é nosso objetivo advogar um modelo conceitual único e completo para a usabilidade. O nosso objetivo também não é a especificação do modelo de interação e de funcionalidade, mas uma complementação a esta especificação que permita-nos distinguir quais dentre os seus componentes determinam o Modelo de Usabilidade. Ela é um *especificação do Modelo de Usabilidade como objeto da Mensagem do Designer*, ou simplemente ***especificação da Mensagem do Designer***. Nosso objetivo é apenas mostrar como os componentes conceituais do Modelo de Usabilidade podem ser estruturados como objeto (conteúdo) da Mensagem do Designer. Esta estruturação semântica fornece um modelo teórico no qual está baseado o sistema semântico do nosso sistema semiótico. Os componentes funcionais e de interação devem ser especificados com as linguagens convencionais e complementados pela especificação do conteúdo da Mensagem do Designer. Nosso meta-modelo para a usabilidade apresenta os componentes num nível abstrato o suficiente para que possamos fazer a integração destes conceitos com aqueles existentes nas linguagens de especificação do modelo funcional e de interação. Apresentamos uma notação que é parte de uma linguagem de especificação da Mensagem do Designer. Mostramos ainda como linguagens e formalimos de especificação do modelo de interação e de funcionalidade podem ser integrados à nossa proposta.

Motivação

A estruturação do Modelo de Usabilidade é motivada pelo modelo do processo de interação apresentado por Norman (introduzido no capítulo 3) que descreve os passos necessários para o usuário realizar o *mapeamento tarefa-ação* [Norman 86]. A nossa escolha justifica-se pela ampla divulgação e aceitação que este modelo teve entre pesquisadores da área. Com isto mostraremos como a nossa abordagem supre algumas das limitações daquele modelo, já discutidas também no capítulo 3.

Norman descreve dois golfos que o usuário deve atravessar quando interage com o sistema. Um deles é o *golfo da execução* que, partindo do *estabelecimento da meta,* compreende as etapas de *formulação da intenção, especificação da seqüência de ações e execução.* O outro é o *golfo da avaliação* que compreende as etapas de *percepção, interpretação e avaliação* que deve permitir avaliar se as modificações do sistema ativadas pelo processo de execução estão de acordo com as metas estabelecidas.

Nosso enfoque no modelo do Norman é principalmente no golfo da execução. Esta restrição é justificada por dois motivos. O primeiro é o fato de que o Modelo de Usabilidade deve, no mínimo, possuir tudo aquilo que é necessário para o usuário desempenhar o *mapeamento tarefa-ação.* Queremos identificar os componentes que são requisitos necessários para cada uma das etapas da execução. O segundo motivo é que na perspectiva de metacomunicação da Engenharia Semiótica os processos de percepção e intepretação não são apenas utilizados para avaliação, mas de maneira muito mais ampla como parte da intepretação da Mensagem do Designer. Nosso modelo mostrará mas adiante como este processo deve ser estendido para abranger tanto a comunicação usuário-sistema quanto a comunicação designer-usuário. A Figura 0-3 ilustra o nosso enfoque. Queremos estruturar o Modelo de Usabilidade identificando os componentes utilizados no processo de interação.

Algumas pequenas modificações no modelo de Norman são feitas. O *estabelecimento da meta* é incluído como parte do processo de mapeamento tarefa-ação. A *formulação da intenção* é colocada aqui como um *planejamento* no qual o usuário precisa decidir o que fazer a partir das funções que o sistema oferece. A *especificação da*

seqüência de ações é chamada de *especificação da estrutura de ações* de maneira a enfatizar que as ações não são necessariamente sequenciais.

As subseções seguintes apresentam os componentes e em seguida mostramos o papel de cada um deles no mapeamento tarefa-ação.

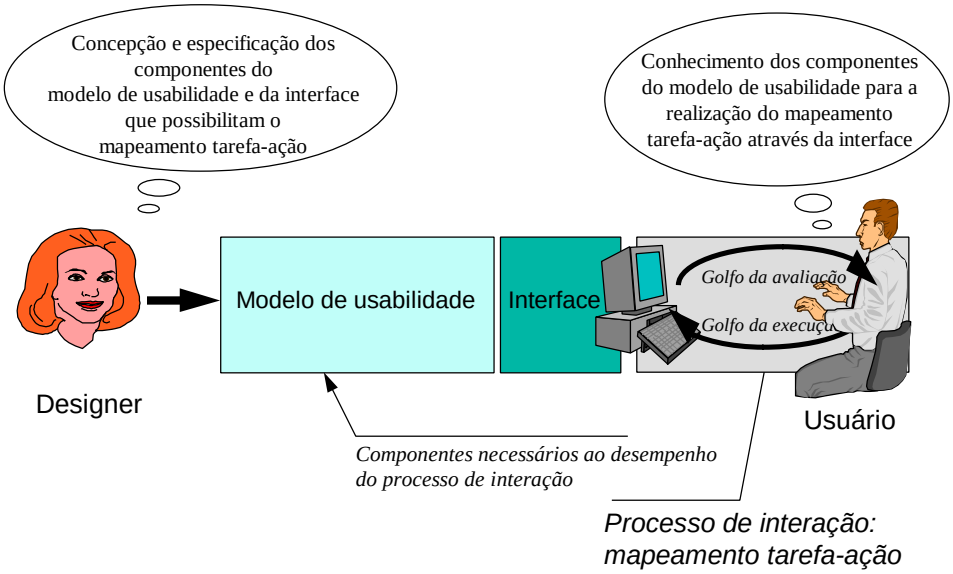


Figura 0-3: O Modelo de Usabilidade e o processo de interação.

Os componentes essenciais do Modelo de Usabilidade

Nesta subseção, descrevemos de maneira sucinta todos os componentes conceituais do nosso modelo. Nas subseções seguintes, vamos abordar cada um destes componentes isoladamente e mostrar como eles devem ser integrados ao processo convencional de especificação funcional e do modelo de interação.

Os componentes que apresentamos são identificados dentre aqueles elementos da especificação funcional e de interação que são essenciais para o usuário atravessar o golfo de execução, ou seja estão voltados para a usabilidade. A primeira distinção conceitual feita dentro do Modelo de Usabilidade é entre os componentes de funcionalidade e os de interação, já discutida desde a introdução. A *funcionalidade* define aquilo que um sistema permite fazer. A *interatividade*, ou, como preferimos, o *modelo de interação*, determina como o usuário pode interagir com as funções e os objetos definidos na funcionalidade.

Os componentes do modelo de interação ainda não são descritos de maneira sistemática na maioria das abordagens para design de interfaces. Um dos motivos é a diversidade de estilos de interação, cada qual com suas particularidades. Nas linguagens de comandos é possível identificar o conceitos de *comando* e associado a ele a sua semântica, sua sintaxe e seu léxico. Nos outros estilos, tal como *menus* ou *preenchimento de campos de formulários*, o conceito de comando praticamente não existe. Na interação por menus, o usuário praticamente faz escolhas entre os elementos de um menu indicando uma opção através de ações atômicas. Na interação por preenchimento de campos, o que o usuário faz é basicamente fornecer dados alfanuméricos através do teclado. Nas interfaces por manipulação direta o usuário realiza suas tarefas através de ações que manipulam representações da aplicação. Os conceitos principais neste caso são *ação* e *representação.* As interfaces GUI/WIMP apresentam todos estes diversos estilos de interação num ambiente integrado.

A abordagem lingüística de *gramática de ações* introduzida por [Reisner 81] e estendida por [Payne & Green 86] permite estruturar de acordo com uma gramática as ações que o usuário realiza numa interface gráfica. Podemos dizer que as sentenças estruturadas de ações são um *comando*. Inspirado nesta idéia, nosso modelo de interação está baseado no conceito de ***comando de função***. Um comando de função é uma estrutura de articulação de ações que pode ser associada a uma função do sistema. Em linhas gerais, este conceito visa determinar quais as ações que são necessárias para controlar uma função e modificar o estado do sistema. Veremos mais adiante que um comando de função é um signo que possui expressão e mensagem. Não utilizamos o termo *signo de comando* para não confudir com o termo *mensagem de comando* (que será visto no próximo capítulo) que também é um signo.

Ainda em relação ao modelo de interação, temos o conceito de ***visualização*** dos estados do sistema que permitem ao usuário avaliar os efeitos de seus comandos de funções e decidir quais as próximas funções a serem comandadas.

A funcionalidade é de fundamental importância para o design de interfaces na abordagem da Engenharia Semiótica uma vez que ela é a parte central daquilo que o designer necessita comunicar para os usuários. Porém, a maioria das abordagens e

ferramentas para o design de interfaces estão restritas ao componente de interação não indicando como eles precisam ser associados aos elementos da funcionalidade.

A ciência da computação apresenta diversos modelos para a funcionalidade de sistema. A maior parte deles visa descrever a funcionalidade como uma abstração do funcionamento, isto é, sem detalhes de implementação. Eles estão baseados nos conceitos de dados e funções (ou procedimentos). Os dados são os elementos destinados a representar informações. As funções determinam como eles podem ser processados. Neste sentido, as linguagens de especificação funcional visam especificar de maneira abstrata todos os componentes do sistema e não apenas aqueles que se destinam à usabilidade.

A funcionalidade voltada para a usabilidade não deve descrever o sistema em termos de dados, tipos de dados, recipientes, fluxos de dados e estados de processadores. O que interessa ao usuário são as soluções para os problemas do domínio. A funcionalidade deve ser descrita em termos de *aplicações de dados a conceitos do domínio*, ou seja, em termos de informações que representam conceitos e fatos de um domínio. Uma informação pode representar conceitos tais como propriedades de um objeto do domínio, classes de objetos, categorias de uma taxonomia, itens de uma tabela, quantidades de objetos, componentes de objetos, etc. Por exemplo, informações de conceitos do domínio podem ser *hora, data, nomes de..., cores, quantias em dinheiro, número de telefones*, e diversas outras.

A idéia de representação de informações referentes a conceitos do domínio é capturada pela noção de *signo*. Na nossa abordagem semiótica para o modelo de funcionalidade as informações do domínio são representadas por ***signos do domínio*** e podem ser modificadas por ***funções aplicadas***. O signo do domínio é uma abstração das diversas formas que uma informação do domínio pode ser representada num sistema. As funções aplicadas são conceitos que se referem aos componentes operacionais que modificam um signo do domínio ou outras funções aplicadas. Mais adiante descreveremos estes componentes com detalhes.

O papel dos componentes no *mapeamento tarefa-ação*

Nesta subseção, vamos mostrar como os ***signos do domínio, funções aplicadas, comandos de função*** e ***visualizações*** são os quatro componentes essenciais para a travessia do golfo da execução. A Figura 0-4 descreve este relacionamento.

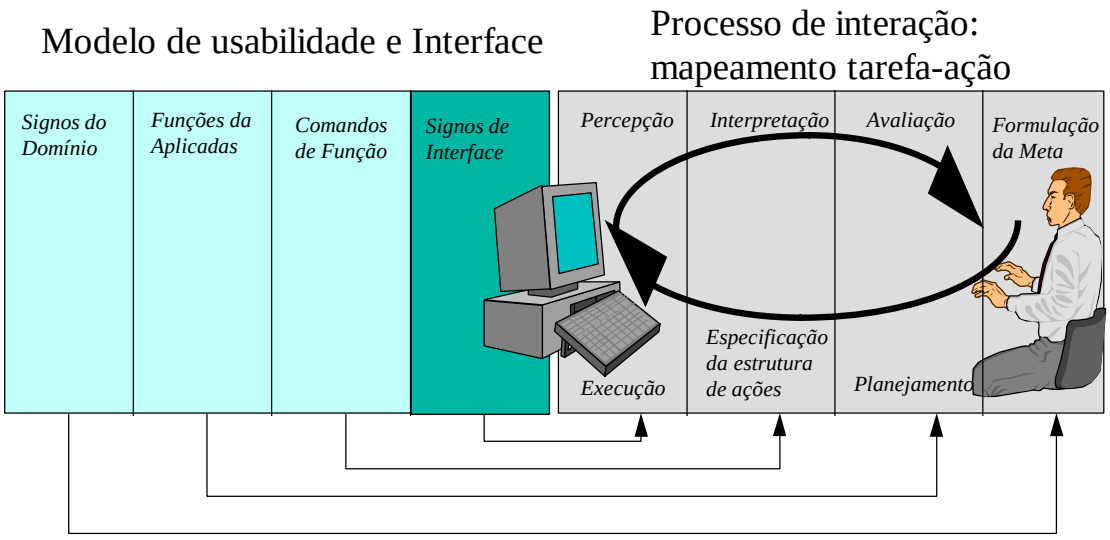


Figura 0-4: O papel dos componentes no mapeamento tarefa-ação.

O estabelecimento da meta do usuário corresponde à mentalização de um estado futuro do sistema. Este processo é motivado por *estados atuais do sistema.* Um estado do sistema é caracterizado pelo estado de seus componentes. O estabelecimento da meta, portanto, corresponde a determinar os futuros estados dos *signos do domínio* e das *funções aplicadas.*

Um *planejamento* consiste na atividade mental de elaborar uma estrutura (plano) de tarefas a ser executada. Neste processo a meta é dividida em sub-metas, sucessivamente, até chegar-se àquelas para as quais existe uma tarefa possível de ser realizada através de um comando de função. No processo de interação com o sistema, o planejamento, chamado por Norman de formulação da intenção¸ consiste no estabelecimento de sub-metas que possam ser atingidas diretamente por funções do sistema (que chamamos de funções aplicadas). Cada função aplicada tranforma o estado de um componente de funcionalidade. Cabe ao usuário escolher a função que produzirá o

estado futuro estabelecido como meta. Este processo de planejamento apenas é possível para os usuários que conhecem quais são as funções oferecidas pelo sistema.

A *especificação da estrutura de ações* visa determinar quais são as ações e de que maneira elas devem ser desempenhadas para que o usuário possa controlar a execução de uma função pelo sistema. Aplicando os conceitos semióticos de semântica e sintaxe ao conceito de *comando de função* introduzido na seção anterior, podemos dizer que a estrutura de ações é a sintaxe de um comando de função e a função aplicada associada é a sua semântica. O usuário precisa conhecer a semântica dos comandos para saber quais os que estão associados às funções que ele planejou e saber a sua sintaxe para especificar quais as ações a serem executadas.

Para que as ações especificadas possam ser executadas faz-se necessário que os objetos de interação, ou *widgets* no caso das interfaces GUI/WIMP, sejam disponibilizados para o usuário. A cada widget devem estar associadas uma ou mais ações que podem ser executadas pelo usuário. *Widgets* podem ser compostos de maneira a permitir que as estruturas de ações possam ser executadas. O nosso modelo para interfaces abstrai os *widgets* através do conceito de *signo de interface* que será discutido na próxima seção.

O processo para avaliar se a meta foi atingida pelo mapeamento tarefa-ação necessita da *visualização* dos estados dos componentes do sistema. Da mesma forma, o estabelecimento de uma nova meta depende da avaliação da situação momentânea para que possa ser iniciado um novo ciclo de ação.

O nosso modelo conceitual para os componentes de interação e funcionalidade do sistema oferece abstrações que podem não abranger todos os casos de interface. O nosso objetivo é enfatizar em distinções conceituais que proporcionem a pesquisadores e designers explicações sobre o processo de design, de interação e a comunicação designer-usuário.

Introduzindo um exemplo

Vamos ilustrar os conceitos que serão apresentados nas próximas subseções com um exemplo de um *sistema de transações bancárias*. O objetivo desta aplicação é permitir aos usuários realizarem consultas e transferências de suas contas bancárias. Os

analistas da equipe de desenvolvimento identificaram os seguintes conceitos do domínio: *cliente, conta, saldo, limite-de-crédito*. Para cada um destes conceitos informações específicas estão associadas. Os clientes do banco são identificados por um *número-do-cliente*. As *contas* são identificadas por *número-da-conta*. Um cliente pode ter várias contas, cada uma com um *saldo* independente. Cada cliente possui um *limite-de-crédito* que se aplica a cada uma das suas conta. Um saldo positivo ou limite de crédito de uma conta não cobre um saldo abaixo do limite em outra.

A análise permitiu a um designer elaborar a seguinte especificação informal. Uma *conta-bancária* descreve informações a respeito de *conta* e de *cliente*. *Conta* é um mapeamento entre um *número-de-conta* e um único *dado-de-conta*. *Cliente* é um mapeamento entre um *número-de-cliente* e um único *limite-de-crédito*. Um *dado-de-conta* é composto pelo *número-de-cliente* e pelo *saldo*. A restrição é que o saldo de cada conta seja sempre maior ou igual ao limite de crédito do cliente.

As funções que manipulam as informações associadas a cada um destes conceitos são as seguintes. A função *consulta de saldo* permite ao usuário verificar o saldo de sua conta. A função *saque* permite ao usuário debitar dinheiro de sua conta e a função *depósito* permite a ele creditar nela. A função *transferência* permite ao usuário transferir valores de suas contas para as de quaisquer outros clientes. Restrições levantadas na análise determinam que cada *função* só pode ser realizada com um única conta do usuário de cada vez. Para as funções *saque, depósito e consulta de saldo* são utilizadas as funções *soma* e *subtração*. Para a função *consulta de saldo* é necessário a função *mostra-saldo*.

Outras soluções de especificação alternativas poderiam também satisfazer aos requisitos descritos na análise. Nas seções seguintes, vamos utilizar esta análise e especificação funcional informal nas discussões a respeito da especificação do Modelo de Usabilidade.

Signos do Domínio

Os signos do domínio são especificados por uma estrutura que descreve de maneira abstrata os componentes do sistema que são utilizados para representar conceitos do domínio de aplicação. Ela é uma abstração das entidades computacionais que devem realizar a codificação das informações relativas aos conceitos de um domínio na forma de

estruturas de dados, quando o objetivo for a implementação, ou de *representações visuais* quando o objetivo for a interação com o usuário. Enquanto estruturas de dados são elementos do escopo da ciência da computação, a representação visual é do escopo da semiótica.

Vamos voltar ao exemplo do sistema de transações bancárias. Na análise daquele sistema foram identificados os diversos conceitos com os quais o usuário pode manipular informações. Para um programador as informações sobre clientes e contas devem poder ser representadas por tipos de dados de uma linguagem de programação. Por exemplo, *número-de-conta, nome-de-cliente* e *limite-de-crédito* podem ser representadas por *inteiros*; *saldo* por *real*; e *conta, cliente e conta-bancária* por estruturas compostas como *struct* (linguagem C) ou *record* (linguagem Pascal).

Para o usuário, entretanto, é necessário especificar qual a melhor representação visual para cada um deles. Em termos de nosso modelo de interface, é preciso especificar qual seria o *signo de interface* mais adequado para esta representação. As informações relativas a *número-de-conta* podem ser representadas por *dígitos numéricos* e *nome-de-cliente* pode ser do tipo *texto*. O *saldo* e o *limite-de-crédito* podem ser representados por um gráfico que indique visualmente através de dimensões, posições e cores, o limite de crédito e o valor do saldo. A Figura ilustra esta representações visuais.

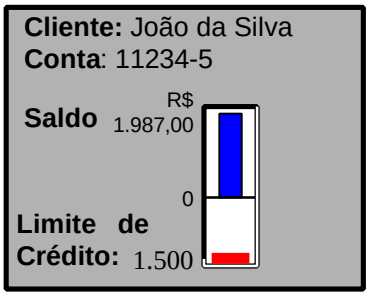


Figura 0-5: A representação visual de uma informação.

Um signo do domínio deve ser associado às outras formas de representação que podem ser utilizadas para as informações associadas aos conceitos do domínio. O tipo de representação computacional (tipo de dados) deve ser decidido pelo especialista em programação. O tipo de representação visual, por sua vez, deve ser decidido por um

especialista em comunicação visual[18]. No nosso meta-modelo conceitual para os componentes da funcionalidade o signo do domínio será representado de maneira abstrata deixando para os especialistas a decisão das representações visuais e computacionais. O objetivo desta representação abstrata é permitir um mapeamento dela com os tipos das representações computacionais e visuais. O tipo de representação abstrata pode ainda ser associado aos tipos normalmente utilizados em especificações funcionais e de interação. No caso de especificações formais estes tipos podem ser os conceitos matemáticos de *conjuntos, seqüências, números naturais ou inteiros, relações e funções*.

A estrutura de signos do domínio é composta pelos seguintes componentes. O ***nome do conceito do domínio*** representa o conceito do domínio que o signo deve representar fazendo a ligação semântica da representação com o domínio. O ***tipo de representação abstrata da informação*** descreve a representação independente de como ela será visualizada pelo usuário ou implementada pelo programador. Podemos denominá-lo também de ***tipo-signo do domínio***[19].

Associada a esta estrutura podem estar os tipos de representação computacional e visual. O ***tipo de representação visual*** da informação especifica a representação visual que será utilizada pelo usuário para obter ou fornecer uma informação ao sistema. Ela será feita através de ***signos de interface***. O ***tipo de representação computacional (tipo de dado)*** especifica a representação computacional que será utilizada para representar a informação usando uma linguagem de programação.

A *estrutura de signos do domínio* completa possui os seguintes componentes:

Nome do conceito do domínio
Tipo da representação abstrata da informação
Tipo do dado da informação
Tipo do signo de interface

[18]Foge ao escopo do nosso trabalho oferecer meios de auxiliar ao designer a escolher qual o melhor tipo de representação visual para representar um certa informação. A representação de informação, embora seja um aspecto essencialmente semiótico, tem sido bastante abordada na literatura [Tufte 90], e não contribui para os aspectos interativos que diferenciam o *medium* computacional do *medium* visual convencional.

[19]O ***estado*** ou ***informação*** do signo do domínio especifica uma instância da representação e é um ***token-signo do domínio***.

Na especificação da Mensagem do Designer que é complementar à especificação tradicional do modelo de funcionidade podemos nos limitar à especificar seu nome e o tipo da representação abstrata. O tipo de dado refere-se a aspectos de implementação que não precisam ser especificados em nossa abordagem semiótica. O tipo de representação visual apenas precisa ser especificado durante a escolha de quais signos de interface utilizar para cada signo do domínio. O capítulo 6 apresenta regras que associam os tipos de representação aos *widgets*, sem no entanto, entrar em detalhes sobre aspectos de visualização de informação (ver nota 1).

Vamos especificar, como exemplo, os conceitos de *número-da-conta* e *nome-de-cliente* de acordo com a especificação informal do sistema de transações bancárias introduzido anteriormente.

Nome do conceito do domínio: **Número da Conta**
Tipo da representação abstrata da informação: **Numérico**
Tipo do dado da informação: **Inteiro**
Tipo do signo de interface: **Digitos numéricos**

Nome do conceito do domínio: **Nome de Cliente**
Tipo da representação abstrata da informação: **Texto**
Tipo do dado da informação: **String**
Tipo do signo de interface: **Texto**

Vamos introduzir aqui uma notação para a descrição dos componentes do Modelo de Usabilidade de usabilidade complementar. Esta notação faz parte da linguagem de especificação[20] que será apresentada no próximo capítulo.

Domain-sign *<domain-concept-name>* **representation-type** *<representation-type>*

Usando esta notação *número-da-conta* e *nome-de-cliente* podem ser especificados da seguinte forma:

```
Domain-sign Número da Conta representation-type Numérico
Domain-sign Nome de Cliente representation-type Texto
```

Existem diversos tipos de representação que podem ser utilizados em signos do domínio. Vamos propor um conjunto reduzido de tipos de representações abstratas que possam ser mapeados na maioria dos tipos de dados e *widgets* de interfaces. Um tipo de

representação abstrata (tipo-signo do domínio) será definido como um conjunto das diversas representações possiveís. Uma informação (uma instância do tipo) é um elemento deste conjunto de representações.

Os tipos de representações são divididos em simples e estruturados. Os tipos estruturados são definidos em termos de tipos simples.

Tipos Simples:

valores -verdade
numéricos
textos

Os tipos valores-verdade são binários, como sim/não ou liga/desliga e correspondem aos tipos *booleanos* da maioria das linguagens de programação. Os tipos *numéricos* correspondem aos tipos *inteiro* ou *real*. O tipo *texto* corresponde ao conceito de *string* (cadeia de caracteres) nas linguagens de programação.

Os tipos simples podem ainda ser classificados em tipos mais específicos de acordo com parâmetros de variação. Os tipos *numérico* e *texto* podem variar em termos de ***tamanho das instâncias*** *(grande ou pequeno)* de ***quantidade de instâncias do tipo*** (*muitas* e *poucas*) e ***limite da quantidade de instâncias*** (*finito ou infinito*). Por exemplo, o conceito *dias-do-mês* pode ser representado pelo tipo *numérico* que varia num intervalo *finito* com *poucas* instâncias de tamanho *pequeno* (entre 0 e 31). Já o conceito *saldo* deveria ser representado por números num intervalo *infinito* com *muitas* instâncias de tamanho *grande*. Estas especificidades são importantes para a escolha do tipo de representação visual e computacional adequadas.

O tipo *texto* que é uma seqüência indefinida de representações alfanuméricas pode ser restringido para um conjunto de instâncias do domínio pré-definidas (em tempo de design). Neste caso, o signo do domínio deste tipo pode representar um *conjunto fechado* de informações. No outro caso o conjunto é *aberto* e definido ao longo do uso da aplicação. Um exemplo de tipo *aberto* é a representação de *nomes-de-pessoa* enquanto que nomes de *estados-do-Brasil* pode ser um tipo texto *fechado*.

Tipos Estruturados:

[20]Os termos da linguagem estão em inglês, como será justificado no próximo capítulo.

conjuntos
tabelas
hierarquias
redes

Os tipos estruturados são descritos em termos de outros tipos. Eles podem ser *homogêneos*, quando a estrutura é de um mesmo tipo apenas, ou *heterogêneos*, quando formados a partir de tipos distintos. Cada um destes tipos serão descritos pelos termos *conjunto-de, tabela-de, hierarquia-de, rede-de*, seguido pelo nome do tipo. Como nosso objetivo é apenas ilustrar o conceito de *signo do domínio*, não aprofundaremos numa classificação exaustiva dos diversos tipos.

O saldo de um cliente ao longo de um período de tempo (um extrato da conta) poderia ser representado em signo do domínio do tipo *tabela*. A representação computacional poderia ser do tipo (em C) *array* de *struct* e a representação visual poderia ser um *gráfico de barras*, como descrito na Figura 0-6. A estrutura do signo do domínio seria:

Nome do conceito do domínio: **Extrato**
Tipo da representação abstrata da informação: **Tabela-de numérico**
Tipo do dado da informação: **Struct { ... } nome[...];**
Tipo do signo de interface: **Gráfico de barras**

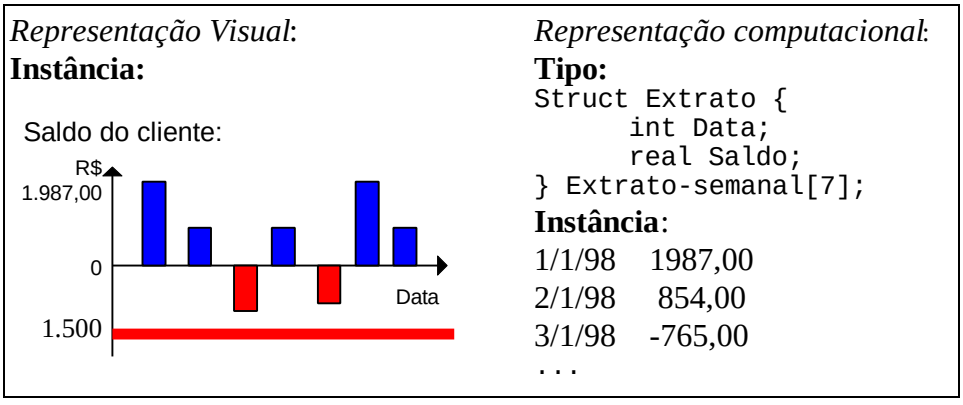


Figura 0-6: Representações do signo do domínio *Extrato*.

Para finalizar é preciso introduzir dois últimos conceitos relacionados com *signos do domínio*. Um deles é o de ***objeto de representação*** que refere-se ao *medium* no qual as instâncias dos tipos de representação são veiculadas. Nas representações computacionais este *medium* é a uma *posição de memória*, representada nas linguagens de programação

pela noção de *variável.* Nas representações visuais em interfaces gráficas este *medium* é um *widget*. Um *campo de texto,* por exemplo é um *widget* que veicula representações do tipo *texto.*

O outro conceito é o de ***estado*** de um signo do domínio que corresponde à informação que cada *objeto* representa num determinado instante. Um objeto de representação *variável* possui diferentes *estados,* ao passo que um objeto de representação *constante* permanece num *único estado.* O *estado* de um objeto de representação é uma instância do tipo de representação. O estado é visual, no caso da interface, e computacional, no programa. Como o estado de um objeto de representação refere-se à informação representada pelo signo do domínio, podemos referenciá-lo apenas por ***informação***.

Funções Aplicadas

A funcionalidade do sistema caracteriza-se pelas ***funções aplicadas*** que operam sobre objetos de representação, modificando seus estados e, conseqüentemente, as informações dos conceitos do domínio. As funções aplicadas são apenas aquelas que, dentre todas as funções de um sistema, aplicam-se às tarefas do usuário. Elas constituem as soluções-em-potencial que o designer concebe e comunica ao usuário.

Na especificação da Mensagem do Designer a funcionalidade deve ser definida pelas *funções aplicadas* disponibilizadas para os usuário de acordo com as suas necessidades e deve ser complementada pela especificação das outras funções que serão necessárias para o funcionamento do sistema. No exemplo do sistema de transações bancárias, introduzido anteriormente, as funções aplicadas são *consulta de saldo, saque, depósito e transferência.* As funções *soma* e *subtração* são funções auxiliares que nunca são utilizadas diretamente pelos usuários.

A especificação da função aplicada *transferência* precisa ser complementada pela especificação das funções *subtração* e *soma*. Estas funções não são disponibilizadas para o usuário e, portanto, não fazem parte do Modelo de Usabilidade, apenas do modelo funcional do sistema. Elas são chamadas no nosso modelo de *funções internas* ou *auxiliares.*

A especificação do componente de funcionalidade (que faz parte do Modelo de Usabilidade) da Mensagem do Designer precisa ser complementada por uma especificação funcional tradicional na qual todas as funções do sistema devem ser descritas. Não nos interessa qual das duas deve ser especificada primeiro. A especificação da funcionalidade pode ser elaborada a partir da especificação completa quando, auxiliado por modelos do usuário e da tarefa, o designer identifica quais dentre todas as funções devem ser as funções aplicadas. No caso contrário, o designer pode elaborar inicialmente a especificação das *funções aplicadas* e, em seguida, complementá-la pela especificação completa.

Uma *função aplicada* deve ser especificada como uma transformação de *estados* dos *objetos de representação* dos *signos do domínio* descrita em termos de ***operandos, pré-condições, pós-condições*** e ***controle operacional.***. Os ***operandos*** são os objetos de representação dos signos do domínio utilizados na função. A ***pré-condição*** descreve *estados anteriores* dos *operandos* envolvidos, isto é antes da ativação da função. A ***pós-condição*** descreve o *estados finais* dos operandos, após a execução ter sido completada. Cada função aplicada deve poder ser controlada pelos usuários. O ***controle operacional da função*** é o elemento da especificação que determina quais os tipos de controle que serão disponibilizados para os usuários. Os tipos de controle são: *iniciar, terminar, suspender, continuar, executar passo-a-passo e cancelar.* Na maioria das aplicações apenas os controles de *iniciar e cancelar* estão disponíveis.

Por exemplo, a especificação de uma função aplicada que permite ao usuário efetuar uma transferência de dinheiro entre suas contas deve determinar como pós-condição que o saldo da conta debitada é igual ao valor do saldo inicial subtraído do valor transferido e o saldo da conta creditada é igual ao valor do seu saldo inicial somado ao valor transferido. A especificação das *pré* e *pós-condições* de uma função requer uma notação específica e precisa que foge ao escopo do propósito comunicativo do designer. Deixamos em aberto este aspecto para ser complementado por notações formais da especificação funcional completa, como por exemplo as linguagens *VDM e Z.*

A especificação de uma *função aplicada* na Mensagem do Designer requer a seguinte estrutura:

Nome da função aplicada - *nome que descreve a aplicação da função.*

Operandos - *são objetos de representação dos signos do domínio utilizados pela função.*

Pré-condição - *estados anteriores dos operandos.*

Pós-condição - *estados posteriores de operandos.*

Controle operacional da função - *define as opções de controle do decurso da função: iniciar, terminar, suspender, continuar, executar passo-a-passo e cancelar.*

Vejamos como exemplo a especificação das funções aplicadas do sistema de transação bancária. Vamos nos limitar às funções de *consulta de saldo* e *transferência.*

Consulta de saldo:

Nome: **Consulta de saldo**.

Operandos - **nome-do-cliente, número-da-conta, saldo**

Pré-condição - *a ser especificado*

Pós-condição - *a ser especificado.*

Controle operacional da função: **Iniciar, Cancelar**

Tranferência:

Nome: **Transferência**.

Operandos - **nome-do-cliente-debitado, número-da-conta-debitada, nome-do-cliente-creditado, número-da-conta-creditada, valor-transferido**

Pré-condição - *a ser especificada.*

Pós-condição - *a ser especificada.*

Controle operacional da função: **Iniciar, Terminar**

Cada um dos operandos é um objeto de representação de um signo do domínio já especificado. *Nome-do-cliente-debitado* e *nome-do-cliente-debitado,* por exemplo, são objetos de representação do mesmo signo do domínio (*nome-de-cliente*).

Vejamos agora um outro exemplo que ressalta a importância dos conceitos de *signo do domínio* e *função aplicada.*

Nome: **Calcula Lucro**

Operandos: **Valor de compra, Valor de venda, Lucro**

Pre-condição: **Valor de compra, Valor de venda must have an input value**

Post-condição: **Lucro = Valor de venda - Valor de compra**

Control **Iniciar**

A especificação acima deve ser implementada por um programa que realize uma subtração. Valendo-se do exemplo do programa subtração discutido no capítulo 4,

podemos ressaltar o papel da especificação voltada para usabilidade. Naquele exemplo, mostramos a importância das mensagens de interação através das quais o usuário interpretaria a funcionalidade e o modelo de interação. Com a utilização de signos do domínio mais adequados pode-se aplicar mais apropriadamente um programa de mesmo funcionamento às tarefas do usuário. Os signos do domínio *valor de compra, de venda e lucro* aplicam o programa subtração mais apropriadamente a terefas de vendas. Da mesma forma utilizando signos do domínio como *Número de conta, Valor debitado e Saldo* podemos especificar uma função aplicada *Saque* para o domínio de transações bancárias.

O Modelo de Interação

O modelo de interação visa definir quais as ***interações básicas*** que o usuário precisa desempenhar para construir os ***comandos de funções*** que acionam as *funções aplicadas* do sistema e como os *signos do domínio* podem ser ***visualizados***. Uma ***interação básica*** é a ação que o usuário desempenha com algum *objeto de interação* (um *widget*), possivelmente com o auxílio de uma ferramenta de acionamento. Um ***comando de função*** é formado por uma ou mais *interações básicas* que produzam como efeito uma atividade operacional do sistema (a ativação de uma função de aplicação) modificando estados de seus objetos. Uma ***estrutura de articulação de interações*** descreve a maneira como as *interações básicas* podem ser realizadas de modo a formar um comando.

A especificação do componente de interação da Mensagem do Designer deve ser independente dos objetos de interface. A especificação da estrutura do comando deve ser feita em termos de *interações básicas abstratas* sem entrar em detalhes sobre quais devem ser os *widgets* ou dispositivos de interface, nem quais devem ser as rotinas de software para cada uma delas. Os tipos de *interação básica abstrata* podem ser um *acionamento, fornecimento de caracteres alfanuméricos, uma visualização* ou *uma seleção* que correspondem as atividades desempenhadas na maioria das interfaces.

Um *acionamento* abstrai as ações de pressionar uma única tecla, um clique com o mouse, ou ainda um toque na tela. O *fornecimento de caracteres* corresponde às atividades de digitação das teclas alfanuméricas. A *visualização* corresponde às atividades de perceber e interpretar os diversos signos apresentados na interface. A

seleção é a atividade de interação composta pela visualização de um grupo de opções, por uma tomada-de-decisão e por um acionamento da opção selecionada. É importante observar que ela inclui ações dos componentes perceptuais, motores e cognitivos e está de acordo com o modelo *GOMS* [Card, Moran & Newell 83] mencionado no capítulo 3.

As *interações básicas* de um comando devem ser *ordenadas* por uma das seguintes *estruturas de articulação de interações*, também chamadas de *estrutura sintática do comando*: ***seqüência, agrupamento, combinação, repetição*** e ***seleção***. A estrutura *seqüência* determina que as interações de um comando sejam realizadas numa ordem sequencial específica. No *agrupamento*, as interações podem ser desordenadas, isto é, não precisam ser numa seqüência específica. A *combinação* requer que duas ou mais interações sejam realizadas de maneira dependente uma da outra ou de maneira síncrona. Na *repetição* as interações devem ser realizadas de maneira repetitiva. A *seleção* não é uma estrutura de ordenação, mas uma estrutura de alternativas de interações.

Um comando de função é um signo e como tal possui uma ***expressão*** e um ***significado.*** A *estrutura de interações* caracteriza a ***expressão*** de cada comando e é especificada pela ***sintaxe*** do comando. A interpretação de um comando pelo software da interface causa uma modificação no sistema como, por exemplo, a ativação de uma função. Este efeito que o comando produz é o seu ***significado*** que é especificado pela ***semântica*** do comando. Um comando de função é definido pela especificação de sua sintaxe e de sua semântica.

A especificação de um comando pode ser feita segundo o seguinte esquema:

> Nome-do-comando - *descrição do comando em termos da função aplicada e seus operandos*
> Sintaxe: *estrutura de articulação das interações*
> Semântica: *associação da sintaxe com o controle operacional da função aplicada*

Por exemplo, o comando para a função aplicada *Consulta saldo* pode ser controlada pelo comando:

Nome-do-comando: **Consultar saldo**

Sintaxe: **Seqüência de { *forneça** nome do cliente**, forneça** número da conta, **acione** iniciar, **visualize** saldo***}**

Semântica*: após o acionamento de iniciar* a *função da aplicação* ***consulta saldo*** é ativada e o resultado é representado pelo *signo do domínio* ***saldo***.

A sintaxe deve ser especificada por algum formalismo específico que permita descrever as estruturas de articulação de interações. A semântica é determinada pelos programas da interface que associam as interações com os *controles de funções da aplicação.* Vamos mostrar como formalismos baseados em *redes de transição* permitem a especificação destas estruturas e revelam aspectos de complexidade e computabilidade associados à estruturas.

A especificação das estruturas de interação

As estruturas de articulação de interação permitem definir a sintaxe de um comando como uma *gramática de ação.* A estruturação gramatical oferece as vantagens de avaliar o potencial expressivo e o grau de complexidade da sintaxe do comando. A complexidade máxima das estruturas de articulação está limitada à capacidade de processamento computacional e à capacidade cognitiva dos usuários.

A capacidade cognitiva é objeto de estudo das abordagens cognitivas como as vistas no capítulo 3. Diversos formalismos das abordagens cognitivas permitem avaliar questões de complexidade e consistência das estruturas de articulação das ações, tais como *TAG (Task-Action Grammar)* [Payne & Green 86] ou PROCOPE [Poitrenaud 95].

A complexidade computacional fornece-nos constatações interessantes e reveladoras sobre o impacto da complexidade de articulação na usabilidade. Baseados da *Hierarquia de Chomsky*, vamos discutir os níveis de complexidade de articulações de acordo com a sua classificação das gramáticas em *regulares, livres de contexto, sensíveis a contexto* e *irrestritas* [Chomsky 65]. De acordo com a teoria da computação, uma *Máquina de Turing* tem o potencial máximo de processar gramáticas tão complexas quanto as *irrestritas* [Aho et al. 86]. Vejamos agora os formalismos de redes de transição permitem associar estruturas de interação com a hierarquia de Chomsky.

Uma *Rede de Transição de Estados* é um formalismo com potencial equivalente ao das gramáticas regulares e é definido por:

um conjunto finito de *estados* **S**;
um *estado inicial* $\mathbf{S_0}$;
um subconjunto $\mathbf{F} \subseteq \mathbf{S}$ de *estados finais*;
um conjunto finito de *símbolos*;
um conjunto de regras de transição entre estados disparada pela leitura dos símbolos.

Adaptando-as para a especificação de estruturas de interações precisamos considerar que os símbolos lidos são ações (**a)** individuais de um usuário e que os estados finais atingidos por uma série de transições correspondem a *controles de comandos* (**c**) que disparam funções internas da aplicação.

Dentre as estruturas de articulação regulares podemos indentificar alguns padrões que poderão ser associados a configurações de signos de interface (*widgets*). Uma estrutura elementar poderia ser descrita pela rede da Figura 0-7

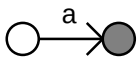


Figura 0-7: Estrutura elementar

Quando o usuário realiza a ação **a** no estado inicial, o software da interface imediatamente transita para o estado final e a *função aplicada* a ele associada é ativada. Toda vez que esta função é mentalmente percebida como uma "tarefa" do domínio de aplicação, por exemplo, a distância articulatória é virtualmente nula.

Estruturas com apenas dois estados ambos finais (ver Figura 0-8) pemite o controle binário de funções da aplicação e determina uma gramática de ação que indica que o usuário pode comandar as funções aplicada c1 e c2 alternadamente.

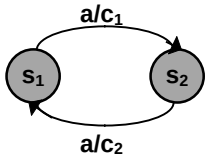


Figura 0-8: Estruturas binárias

Estruturas em seqüência linear, cujo padrão de rede está descrito na Figura (a) são aquelas cujo estado final é atingido após uma seqüência de ações $\mathbf{a_i}$ passando

necessariamente por estados intermediários. Associado ao estado final está um comando de função. Quando todas as ações são idênticas, estamos diante de um caso de repetição finita, como mostrado na Figura (b).

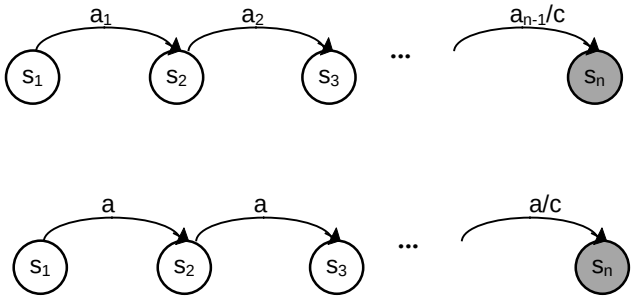


Figura 0-9: Estruturas Seqüência e Repetição

A estrutura *seleção*, cujo padrão de rede está mostrado na Figura 0-10, é aquela na qual um estado final significativo é atingido por uma variedade de ações alternativas. Porém, esta estrutura como está implica apenas que as ações alternativas são "*sinônimos*" para um mesmo comando de função.

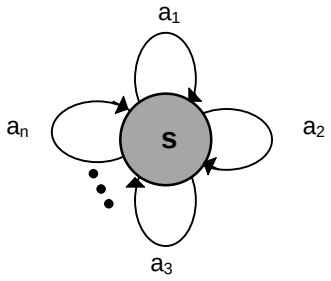


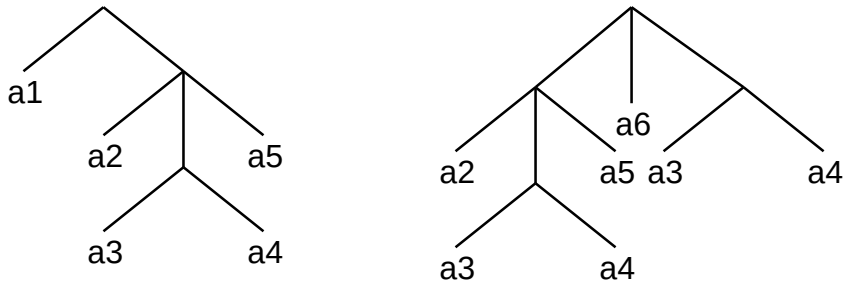
Figura 0-10: Estrutura de Seleção

Os casos mais interessantes derivam porém de uma associação, a cada leitura, de um *contexto* distinto, por exemplo, que irá modular diferencialmente o estado final atingido através de ações de atribuição de características específicas. Neste caso, abandonamos o âmbito das linguagens regulares e entramos no das sensíveis a contexto, para as quais as *Redes de Transição Aumentadas* (*Augmented Transition Networks*, ou ATN) propostas por Woods em 1970, são mecanismos de especificação suficientes.

Entre as *Redes de Transição de Estados* e as ATN's, há a alternativa das *Redes de Transição Recursivas*, através das quais se pode especificar estruturas gramáticas *livres de contexto*. As linguagens ou códigos associadas a este tipo de gramática se caracterizam

por poder estruturar simultaneamente diversas composições (ou sub-estruturas) de ações de cujo estado final significativo depende do completo e correto percurso de transições das estruturas de ações envolvidas. Em termos expressivos relevantes para a metacomunicação designer-usuário, é importante salientar que é somente a partir do nível das linguagens livres de contexto (as que se seguem às regulares e antecedem as sensíveis a contexto na hierarquia de Chomsky) que se podem expressar signos estruturados de interface associados a planos de ação.

Para visualizar este ponto, basta invocar a representação em árvore de duas estruturas representativas de uma linguagem livre de contexto, como por exemplo:



O que se percebe nos dois padrões acima é a recorrência estrutural do padrão a3,a4 e de um segundo padrão que o engloba (a2,a3,a4,a5) dentro de "sentenças" formadoras de comandos bastante distintas. Se estas sentenças têm como semântica a função associada ao estado final válido de uma série de transições em redes recursivas, percebe-se que dentro da aparente "seqüência" de ações de interface, há de fato sub-padrões sintáticos cuja resolução satisfatória é parte integrante de uma resolução satisfatória de maior ordem. Isto está associado à noção básica de divisão de problemas em sub-problemas e do planejamento de soluções dos mesmos. Exemplos interessantes de widgets de interface assim estruturados são os resultantes da Gravação de Macros em diversas aplicações de computação pessoal, tais como editores de texto e planilhas.

A conclusão sobre os paralelos traçados entre as linguagens formais da hierarquia de Chomsky e os padrões de estruturação de interações de maior complexidade é a de que um designer pode valer-se destes recursos para registrar seus padrões de design comunicativo. Todas as propostas de gramáticas de ação, das quais as TAG's são um bom exemplo, valem-se mais ou menos explicitamente desta associação.

O papel da nossa abordagem está em como comunicar estas estruturas gramaticais. A Engenharia Semiótica deve mostrar como construir expressões que indique a estrutura segundo a qual o usuário deve desempenhar as interações. A especificação da Mensagem do Designer deve complementar a especificação da interação indicando como cada *interação* e *estrutura de articulação* deve ser comunicada ao usuário. Mais adiante mostraremos com os *signos de interface* e suas possíveis configurações podem desempenhar esta *função metacomunicativa.*

Visão geral do Modelo de Usabilidade

Iniciamos esta seção com o objetivo de descrever os componentes do modelo de usabilidade como objeto da Mensagem do Designer. Este modelo tem por objetivo identificar quais dentre os componentes de um sistema, descritos pela especificação funcional tradicional da engenharia de software e pela especificação dos modos de interação propostos pela área de IHC, são aplicados à usabilidade.

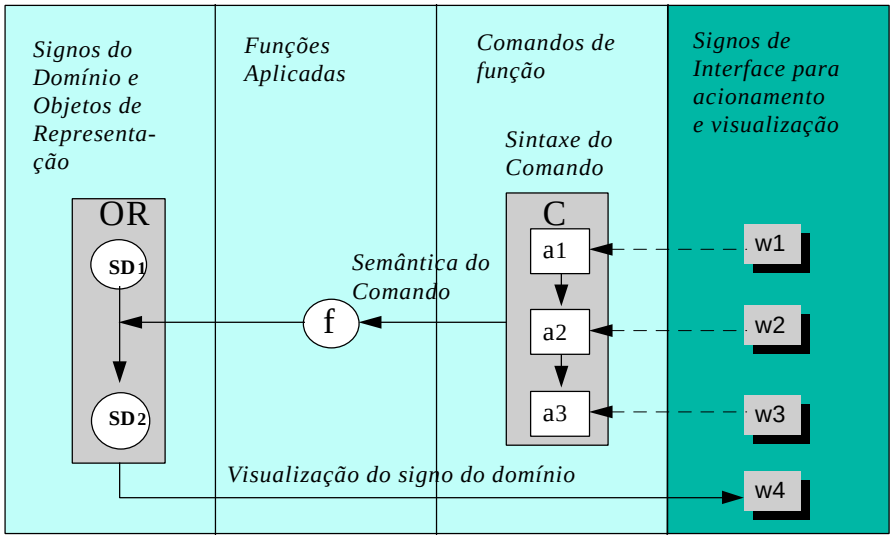


Figura 0-11: O Modelo de Usabilidade e a Interface de Usuário

Na Figura 0-11, estão ilustrados os componentes essenciais do Modelo de Usabilidade conceitual associados aos componentes da interface. Informações de conceitos do domínio são representadas por *signos do domínio* (**SD**). Os *objetos de representação* (**OR**) são entidades computacionais que funcionam como *medium* para os *signos do domínio*. O estado de cada OR expressa signos do domínio diferentes e devem ser visualizados por *signos de interface* (*widgets* - **W**) apropriados (ver seção 5.3). Estes

estados são modificados por *funções aplicadas* (**f**) que são aquelas funções do sistema que podem ser comandadas pelo usuário e que alteram estados do sistema. Estados de funções aplicadas também devem ser comunicados ao usuário, embora não estejam aparecendo nesta figura. Um *comando de função* (**C**) associa uma estrutura de ações do usuário (**a**) ao controle operacional de uma função aplicada. A estrutura de articulação de ações é a sintaxe do comando. Exemplos de estruturas que podem ser especificadas por formalismos equivalentes a Redes de Transição de Estados são *seqüência, uma seleção e um repetição.* Nesta figura está ilustrada uma *seqüência* das ações (**a_1, a_2** e **a_3**). A *semântica* de um *comando de função* (C) associa uma determinada estrutura de articulação de ações a uma *função aplicada* (**f**). Cada ação deve ser desepenhada com um *signo de interface* (**W**).

Na maioria das interfaces GUI/WIMP diversos acionamentos do usuário não constituem *comandos de função* uma vez que eles não alteram estados de componentes funcionais, mas alteram apenas estados de elementos da interface, como maximizar ou minimizar. Ações que atuam no controle da interface são *comandos de controle* da apresentação dos elementos de interface. Eles são chamados de *controles da leitura da Mensagem do Designer*. Este caso não está ilustrado na figura anterior e será discutido na próxima seção.

Mentalmente, com relação ao domínio de aplicação, o usuário tipicamente persegue determinadas *metas* a serem atingidas. Estas metas podem ser concebidas em diferentes níveis (e.g. a meta de se *escrever um memorando* ou a de *formatar um parágrafo*, numa aplicação de edição de textos). Quando concebidas em níveis macroscópicos (como é o caso de escrever um memorando), a existência de um planejamento, formulado em termos de uma organização de tarefas, é tanto mais provável e necessária quanto maior a complexidade das metas. Já quando em níveis microscópicos (como formatar um parágrafo ou mesmo negritar um ou mais caracteres), o planejamento é virtualmente desprezível ou até inexistente, fazendo com que as distinções entre *meta* e *tarefa* se neutralizem.

A interface de usuário, como já definimos no capítulo 2, é a parte do sistema computacional com a qual o usuário entra em contato sensorial. Ela apresenta dispositivos que de acordo com regras ou protocolos previamente definidos permitem a interação entre usuários e sistemas.

Tradicionalmente os modelos propostos para dar conta da interação entre usuário e sistema têm componentes equivalentes aos que, em nosso modelo, estão associados às funções de *acionamento* (genericamente a ENTRADA) e *revelação* (genericamente a SAIDA). A diferença, para nós, reside na função *metacomunicativa,* que se coloca como uma função de segunda ordem sobre as duas anteriores, através da qual um designer comunica explícita (e.g. via tutoriais) ou implicitamente (e.g. via padrões sistemáticos de interação e representação) o seu raciocínio e a sua concepção para os usuários de seu aplicativo. A função metacomunicativa pode ser *direta*, quando o designer envia uma mensagem diretamente através da interface utilizando um signo cuja função seja exclusivamente esta, ou *indireta*, quando o designer envia suas mensagens através de objetos (widgets ou teclas) que têm primordiamente função de acionamento ou de revelação e que, nesta situação, adquirem a função de signo.

O design de interfaces de usuário é um processo comunicativo porque ele se utiliza da interface e do processo de interação que ocorre através dela como *medium* para a elaboração da expressão do signo. Para que o designer possa estabelecer a comunicação com o usuário ele deve produzir distinções na interface que sejam a expressão da sua mensagem. Como a interface tem um potencial ilimitado para a produção de signos, é preciso que o designer disponha de tipos-signos para restringir os interpretantes do usuário "em torno" do Modelo de Usabilidade concebido.

Antes de abordarmos as regras que associam expressões a objetos (os tipos-signos), é preciso analisar o potencial expressivo da interface e realizar o trabalho de *segmentação deste contínuo expressivo* visando identificar os possíveis tipos expressivos que podem ser utilizados num sistema semiótico de apoio ao design. Com isto estamos mostrando como do componentes de interfaces (*widgets*, por exemplo) podem ser estruturados como parte da expressão da Mensagem do Designer. Esta estruturação

sintática fornece um modelo teórico para elaborarmos um sistema sintático que fará parte do nosso sistema semiótico. Os elementos deste modelo são o *medium inteface, a ferramenta de acionamento* e os *signos de interface.* As subseções seguintes apresenta cada um deles.

O *Medium* Interface

Nosso modelo considera que a interface oferece um *medium* (contínuo expressivo) onde as expressões dos signos componentes da Mensagem do Designer podem ser construídos. Denominamos este contínuo de ***Medium Interface*** (MI). O conceito de ***Medium Interface*** (MI) refere-se ao ambiente físico da interface por onde possam ser veiculados mensagens ou signos da interface. É através dele que os signos são interpretados pelo usuário. O exemplo mais comum de medium é a tela onde são veiculados os signos produzidos pelos gráficos computacionais, mas qualquer dispositivo em que o usuário perceba distinções e as possa interpretar como signos pode funcionar com *medium interface.*

O *medium interface* tem a função de veicular mensagens em qualquer sentido, seja do usuário para o sistema, seja do sistema para o usuário, ou, principalmente, do designer para o usuário. Todos os signos que ocorrem na interface fazem parte da mensagem que o designer envia para o usuário. A maioria das propostas teóricas da interação usuário-sistema e ferramentas existentes para o design das interfaces focalizam as suas atenções para os elementos de entrada e saída do diálogo usuário-sistema, sem considerar os outros signos que acompanham os de entrada e saída e que o próprio diálogo está comunicando algo a respeito do Modelo de Usabilidade. Os rótulos textuais e gráficos, as formas dos *widgets*, as cores, são decisões que o designer toma ao projetar a interface e que adquirirem significados para o usuário.

O teclado e os outros dispositivos também veiculam mensagens para os usuários. A função tradicional do teclado ou do mouse é permitir a entrada de dados e o acionamento de funções aplicadas. No nosso modelo, o teclado e o mouse podem ser vistos ao mesmo tempo como *medium*, como signos, e como ferramentas da produção de signos. Entretanto vamos considerá-los apenas como ferramenta de acionamento, deixando apenas para a tela a função de *medium*.

O *medium* por onde uma mensagem pode ser veiculada possui dimensões que podem ser trabalhadas para a produção das expressões [Moles 58]. Estas dimensões podem ser espaciais e temporais. O *medium interface* possui duas dimensões espaciais e a dimensão temporal. Isto significa que se pode produzir expressões ao longo de um plano e ao longo do tempo. Vários signos (*widgets* ou palavras) podem ser mostrados ao usuário durante um certo tempo e substituído por outros. O controle deste processo de apresentação dos signos é decisão do designer e pode ser baseada no tempo ou no processo de interação com o usuário. Este é um aspecto importante para a elaboração da expressão e voltará a ser abordado mais adiante no capítulo 6.

A Ferramenta de Acionamento

No nosso modelo o conceito de ***ferramenta de acionamento*** refere-se a qualquer dispositivo ou mecanismo que o usuário utiliza para manipular o medium interface e acionar signos de interface. Os braços e dedos do usuário, apontadores de mouse, canetas óticas são exemplos de ferramenta de acionamento. É importante notar que, no nosso modelo, o ícone referente ao apontador do mouse não é um signo de interface nem faz parte do medium interface. Ele é a parte que integra a ferramenta de acionamento com o *medium interface* e com os signos de interface. Esta ferramenta é como uma extensão do mecanismo natural de acionamento dos usuários - os seus dedos e mãos.

A ferramenta de acionamento também pode ser utilizada como elemento para a comunicação designer-usuário. O apontador ou cursor possuem formas que podem ser trabalhadas para comunicar ao usuário o tipo de ação que ele pode fazer com um determinado signo de interface. Este aspecto, embora muito importante para a Engenharia Semiótica, não será abordado em nosso trabalho. A justificativa para esta decisão considera que este é um recurso a mais que se pode oferecer para a metacomunicação e que o nosso modelo teórico e as ferramentas a ele associadas devem, nesta etapa inicial, abordar apenas os elementos essenciais.

Os Signos de Interface

Para que possamos elaborar um sistema semiótico de apoio ao design de interfaces precisamos considerar os signos de interface de maneira conceitual e estruturada. Nosso modelo da *interface como expressão* introduz o conceito de ***Signo de***

Interface (SI). Os Signos de Interface dizem respeito a qualquer distinção simbólica que adquira significado para o usuário ou para o designer. Ele representa qualquer elemento, articulado ou não, que pode ser veiculado no medium interface tais como *widgets*, ícones, palavras, teclas, LEDs, menus, caixas de diálogo, assistentes de tarefas, e vários outros. Os resultados de uma computação, os comandos e dados digitados por usuários, os *widgets* de uma aplicação, os arrastos e seleções com o mouse, são todos exemplos de signos da interface de usuário. Um SI é uma abstração teórica sobre entidades computacionais de hardware ou software que podem enviar mensagens para o usuário e/ou podem interpretar comandos ou ações dos usuários. O signo de interface é o componente fundamental de nosso modelo, pois define segmentos que podem ser trabalhados como potenciais unidades expressivas (tipos expressivos). Nas interfaces gráficas os *widgets* como *botões de comando, botões de opções, rótulos, caixas de texto, caixas de diálogo* são exemplos de signos de interface mais comuns.

Os SIs conduzem tanto as mensagens da interação usuário-sistema como também são utilizados para veicular à mensagem do projetista para o usuário, característica que atende aos requisitos de metacomunicação da Engenharia Semiótica. Um SI oferece também a possibilidade de comunicar ao usuário quais ações podem ou devem ser feitas (o modelo de interação) e as funções relacionadas que o sistema deve executar (a funcionalidade). Por exemplo, o widget *botão de opção* comunica opção de uma aplicação está ativada e como o usuário pode modificar (ativar/desativar) cada atributo. Um SI pode ser composto por outros signos e devem poder referenciar-se uns aos outros. A Mensagem do Designer emerge a partir destes signos.

Os signos de interface podem ter função de *acionamento, revelação ou metacomunicação.* A função de acionamento ocorre quando o signo permite que o usuário realize alguma ação que causa uma mudança no sistema. A função de revelação é utilizada para expressar os estados do sistema. A função de metacomunicação ocorre quando o usuário se utiliza das outras duas para enviar a sua mensagem para o designer. Isto pode ser feito pela aparência e comportamento dos signos de interface.

Os signos de interface com função de acionamento definem o repertório de ações do usuário. Por exemplo, o botão de comando é um signo que tem como expressão a

forma de um botão e um rótulo. A forma do botão lembra um botão físico que "convida" o usuário a pressioná-lo da mesma forma que o usuário costuma fazer no mundo físico. Esta forma denota um significado no modelo de interação da aplicação. O rótulo do botão é a parte do signo de interface que denota uma operação na funcionalidade da aplicação. Estas expressões indicam que o usuário deve pressionar o botão para fazer a aplicação executar a respectiva operação da aplicação.

Existem signos de interface que têm apenas função de revelação, isto é, de comunicar apenas um estado ou evento da aplicação sem permitir que o usuário altere sua configuração. Outros revelam apenas mensagens do designer que muitas vezes fazem referência a outros signos de interface. Esta característica satisfaz um importante requisito da Engenharia Semiótica: metacomunicação e referência mútua.

Os signos de interface precisam ser abstraídos em tipos-signos de interface (***Tipos-SI***) para façam parte de um sistema semiótico que dê apoio ao processo de design. Os seus tipos expressivos devem ser formados por padrões de distinções no medium interface. Seu tipo semântico (o significado formal) é denotado pelos elementos correspondentes no Modelo de Usabilidade. Utilizando os tipos-SI de um sistema semiótico, o projetista deve escolher a expressão mais apropriada de maneira a ativar, na mente do usuário, a cadeia de interpretantes (o significado pretendido) que correspondam a elementos da funcionalidade ou do modelo de interação.

O conceito de dispositivos virtuais de interação introduzidos pelas interfaces gráficas - os ***widgets*** - oferece uma possibilidade bastante viável do ponto de vista computacional para a elaboração de sistemas semióticos baseados em tipos-SI. Os *widgets* são elementos gráficos pré-definidos e implementados por programas e ferramentas de interfaces gráficas. Eles constituem tipos expressivos primordiais que o designer deve utilizar para a construção da interface para que possam ser implementados pelos programas computacionais.

No exemplo do GHS, apresentado no capítulo anterior, os signos de interface são construídos utilizando *widgets* do Visual Basic. Para a Engenharia Semiótica eles podem ser vistos como exemplos de signos de interface. Os *rótulos, botões, caixas de texto, listas, quadros, menus e janelas* podem ser os signos de um sistema semiótico que dê

suporte ao design. Entretanto, originalmente, no Visual Basic, os seus *widgets* não foram criados com a idéia de signo de interface da maneira como é proposto na Engenharia Semiótica.

A Figura 0-12 mostra os três elementos do nosso modelo para uma interface GUI/WIMP.

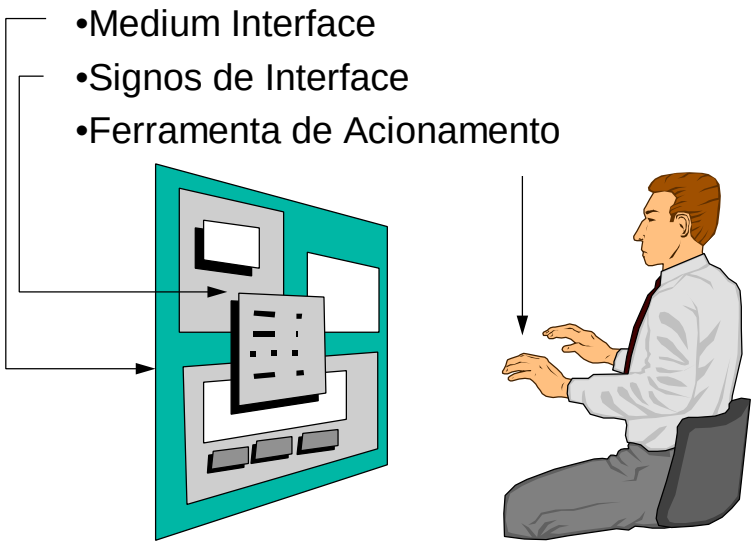


Figura 0-12: A Interface de Usuário como *Expressão*.

Um modelo para os Signos de Interface

Os signos de interface (SIs) são os elementos centrais de nosso modelo. Eles devem ser produzidos a partir de um tipo-signo de interface (Tipo-SI). Um tipo-signo de interface possui um tipo expressivo e um tipo semântico que definem a sua expressão e o seu significado, respectivamente.

Os widgets, e em geral todos os signos de interface, podem ser especificados de tal modo a salientar-se neles o tipo-signo de que derivam (i.e. seu tipo expressivo e seu tipo semântico), tanto quanto as funções de acionamento, revelação ou metacomunicação por eles desempenhadas.

O tipo expressivo é definido através de suas propriedades visuais estáticas e dinâmicas. Estas propriedades definem o que é conhecido por *aparência e comportamento* de um widget (*look and feel*). A aparência é determinada pelas

propriedades visuais do signos de interface. O comportamento é determinado pelo software da interface e oferece *feedback* para o usuário ou representações dinâmicas.

O tipo semântico determina como a expressão é associada aos elementos do Modelo de Usabilidade (as ações que ele permite fazer e o efeito que terá no sistema) e mensagens de metacomunicação do designer. O que define a base do significado de um SI é um programa de computador que interpreta as ações de interface emitidas através de *ferramentas de acionamento* e realiza uma outra ação (seu efeito) que modifica o estado do sistema (da funcionalidade e/ou da própria interface).

Na Engenharia Semiótica, o designer pode realizar sua metacomunicação de maneira direta, usando SIs cuja função seja exclusivamente de metacomunicação, ou de maneira indireta através de SIs com funções de acionamento e revelação. Signos de interface para metacomunicação direta podem ser os widgets *texto, caixas de mensagem* ou *rótulos*, dentre vários outros.

A metacomunicação indireta é desencadeada pela aparência e comportamento dos SIs. Através do encadeamento entre aparência original, aparência de feedback de ativação e a revelação decorrente da interpretação semântica pelo sistema de aplicação, o designer comunica conceitos que devem estar associados a tarefas na mente do usuário. Além disto, a disposição estática e/ou temporal dos signos ao longo de uma interação preenche outra função metacomunicativa, desta vez associada a estruturas de tarefas que deverão estar associadas a planos e metas na mente do usuário.

Vejamos o exemplo de um botão de acionamento tradicional, como os botões 'CANCELAR' encontrados em várias aplicações.

Signo de interface: Botão de acionamento com rótulo “cancelar”

tipo expressivo

aparência:

(I) forma=retangular; cor=cinza; legenda=Cancelar; dimensão 3D

(II) forma=retangular; cor=preta; legenda=Cancelar; dimensão 3D

comportamento:

ao ser pressionado alterna instantaneamente a seqüência de aparências (I), (II), (I).

tipo semântico

traço de usabilidade: execução de operação

interação: ação do usuário=pressionar

funcionalidade: função ou funções correspondente(s) a <nome da operação>

metacomunicação indireta: "se quiser realizar <nome da operação> aperte este botão"

Os Signos de Interface e as estruturas de interação

A aparência e o comportamento dos SIs são fundamentais para a função metacomunicativa. determinar, informalmente, padrões ou estruturas do modelo de interação. Vamos discutir a relação entre o signo de interface e o modelo de interação com o objetivo de verificar quais tipos de estruturas de ações podem ser associadas a cada um deles.

Os dispositivos físicos de interface como teclas, interruptores, e os diversos tipos de botões são bastante limitados em sua capacidade de interação. Os widgets, entretanto, são entidades virtuais construídas por computador. O que define seu significado é um programa de computador que interpreta ações do usuário, a partir de um hardware elementar que podem ser combinadas em grupos de ações mais complexas. Este processo de interpretação das ações permite construirmos estruturas bastante complexas. O limite de complexidade de processamento destas estruturas é aquele determinado por uma *Máquina de Turing*.

Para que a interface possa interpretar estruturas de ações mais complexas é preciso também construir widgets com intepretadores mais complexos ou articulá-los em widgets compostos. Utilizando o conceito de máquina computacional, vamos descrever um modelo para um signo de interface que permita-nos demostrar este seu potencial. Neste modelo, cada signo de interface é descrito por um conjunto de *ações* que o usuário deve desempenhar. Cada ação está associada a uma *mudança de estado* do signo de interface e a um *efeito* no sistema que depende das ações realizadas e dos estados finais. Cada SI possui um *estado inicial* e um ou mais *estados finais*. Esta máquina está ilustrada na Figura 0-13.

Com um modelo para a semântica do SI como uma máquina de estados finitos, podemos representar o seu desempenho por uma *rede de transição de estados*. Considerando o que foi discutido na seção 5.2, dependendo da maneira como esta máquina for projetada, padrões de interação específicos são determinados. Estes padrões constituem as estruturas de articulação das ações dos usuário. Especificando a semântica de um SI através de formalismos equivalentes a uma *rede de transição de estados*, podemos determinar as *gramáticas de ações* que estão associadas a cada um deles. Uma rede de transição permite estruturas de articulação como *seqüência*, *repetição* e *seleção*.

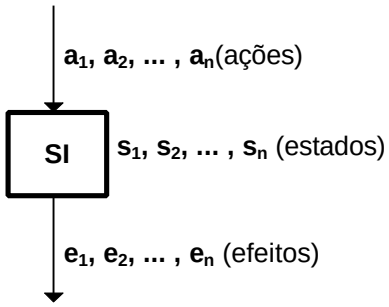


Figura 0-13: Um Signo de Interface como uma Máquina de Estados

A mudança de estados nesta máquina, quando refletida no comportamento e aparência do SI, pode metacomunicar a estrutura de articulação dos acionamentos a ser desempenhada (a sintaxe de um comando) e o efeito que o acionamento terá no sistema (a semântica). Para que isto seja possível é preciso associar o estado do SI com a sua aparência, ou seja, ela deve refletir qual o estado atual do signo. Um signo com um único estado deve ter aparência constante. Um SI com **n** estados deve possuir **n** distinções em suas propriedades expressivas de maneira a possibilitar que ele possua aparências distintas para cada um dos estados.

Este modelo do SI permite-nos mostrar o potencial do conceito de signo de interface e da flexibilidade do modelo de interface. A partir dele mostramos como a interface pode desempenhar a função metacomunicativa através do design de signos de interface que metacomuniquem modelos de interação com estruturas de ações mais complexas. Não é nosso objetivo mostrar como projetar signos de interface. O restante de nosso trabalho irá mostrar como o designer pode realizar a metacomunicação com widgets das principais ferramentas de interface.

A Leitura da Mensagem do Designer

Um sistema interativo pode ter dezenas ou centenas de funções que são oferecidas ao usuário. Estas funções devem ser ativadas por um número de comandos de mesma magnitude. Cada um destes comandos pode ser composto por ações a serem desempenhadas através de *widgets.* A comunicação de todos estes elementos não pode ser feita simultaneamente. O designer deve comunicá-los de maneira organizada e coerente. Os signos de interface que comunicam todos estes elementos do Modelo de Usabilidade precisam então ser estruturados no *medium interface* para que possam ser mostrados ao usuário. Como este *medium* é limitado, é preciso que o designer faça uma *apresentação* das suas mensagens. Sob o ponto de vista do usuário, ele precisa *ler* a Mensagem do Designer. Neste processo de *leitura,* o usuário aciona diversos widgets da interface que não fazem parte de comando de função. Eles são na verdade *comandos de controle da leitura da Mensagem do Designer.*

Por exemplo, o designer pode decidir que um determinado comando seja apresentado em estrutura hierárquica de menus na qual os comandos estão organizados em categorias funcionais. Esta estrutura, além de facilitar a comunicação da funcionalidade, orienta o usuário e diminui o esforço cognitivo para relembrá-la. O designer para acionar este comando precisa *ler* a estrutura de menus que comunica como o designer organizou os comandos para os usuários.

Por exemplo, no MS WORD, para modificar a propriedade *fonte itálico* existe o comando que poderia ser descrito abstratamente como sendo: *set font style = itálico.* Numa interface por comandos, o usuário precisa construir o comando relembrando a sintaxe da linguagem e escrevendo o comando usando um editor de comandos. Nas interfaces WIMP, o comando, seja ele atômico ou estruturado, precisa ser "encontrado" pelo através das estruturas de organização e comunicação escolhidas pelo designer. Na interface do MS Word 6.0, para acionar o comando para modificar estilo do fonte, as ações necessárias são *pressione opção* ***Formatar*** *da barra de menus, escolha e acione a opção* ***Fonte...****, escolha o widget* e realize a ação de *pressionar o estilo* desejado utilizando o widget lista com os estilos disponíveis. As duas primeiras ações não

comandam funções, mas controlam os *widgets* que organizam os comandos. Eles são comandos de interfaces e não de funcionalidades.

Este exemplo revela o fenômeno de comunicação entre o designer e o usuário que não é considerado pelos modelos teóricos, métodos e ferramentas convencionais de design de interfaces. O que ocorre nestas interfaces é que o designer está apresentando para o usuário através de uma estrutura de menus os diversos comandos que o usuário pode acionar. O usuário está realizando o processo de *leitura da Mensagem do Designer.*

Resumo

Nosso modelo teórico estende aqueles propostos pelas abordagens cognitivas apresentados em [Moran 81] e [Norman 86b]. A Engenharia Semiótica, não contesta as abordagens cognitivas, mas apresenta uma perspectiva complementar que visa oferecer fundamentação teórica e metodológica para o design considerando o aspecto de metacomunicação que existe entre o designer e o usuário. Mostramos o papel que o design como produção de signos pode exercer sobre o mapeamento tarefa-ação, *ensinando* o Modelo de Usabilidade para o usuário. Este processo de ensino-aprendizado está ilustrado na figura 0-14 .

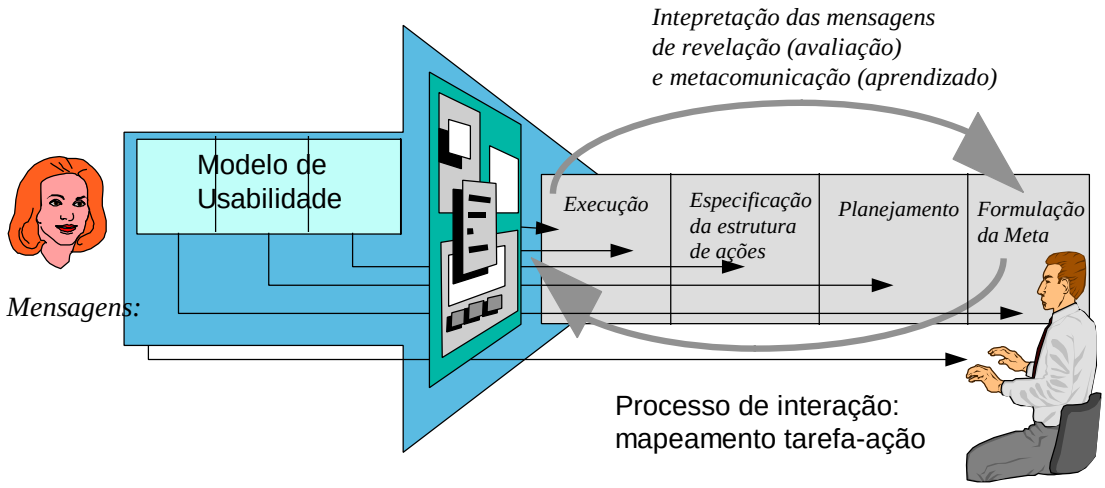


Figura 0-14: A Mensagem do Designer ensinando o Modelo de Usabilidade.

Com a nossa estruturação do Modelo de Usabilidade - segmentação do objeto da mensagem - identificamos diversos elementos que podem ser comunicados ao usuário. Os elementos do modelo teórico para a interface são a base para a construção de tipos-

signos de um sistema semiótico de apoio elaboração da Mensagem do Designer. Tal sistema deverá oferecer ao designer um repertório de tipos-SI (*widgets*) que possam ser aplicados nas diversas unidades de significação do Modelo de Usabilidade. O designer deve construir a expressão da sua mensagem com instâncias (*token*s-SI) dos tipos-SI que possam ser dispostas nas dimensão do medium interfaces em configurações espaciais (layout) e temporais. A Figura 0-15 ilustra nossa abordagem para o design de interfaces de usuário.

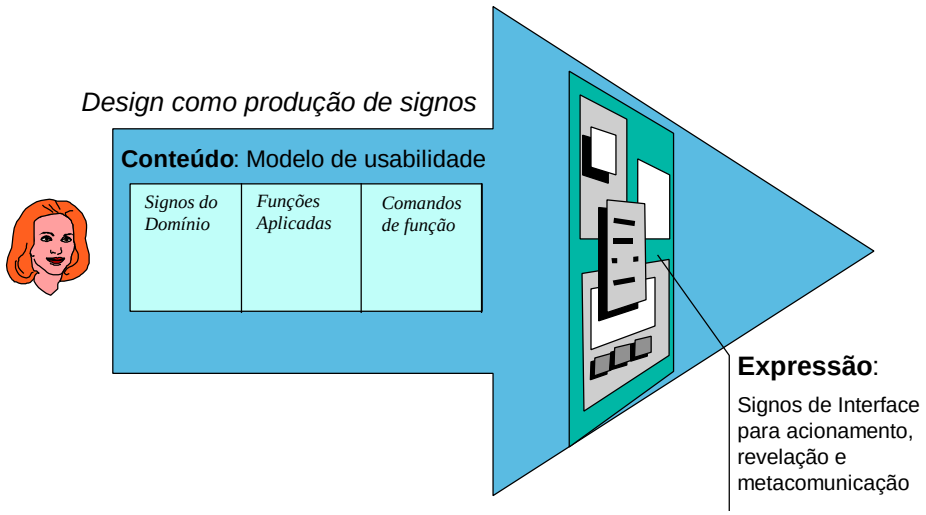


Figura 0-15: O design com produção de signos.

O nosso trabalho não abordará questões referentes ao layout de tela nem aspectos de controle temporal de apresentação. O layout de tela requer o talento e a sensibilidade de designers gráficos que não podem ser capturadas em um sistema semiótico. Este aspecto é comparável aos fatores estilísticos e estéticos que tornam a literatura algo mais do a simples codificação de uma mensagem numa linguagem natural.

Para dar conta destes aspectos um sistema semiótico para o design de interfaces de usuário deveria ser uma Linguagem Visual baseada em computador cujo vocabulário fosse composto por elementos visuais que pudessem ser articulados por regras gramaticais em sentenças visuais. Entretanto, como vimos acima, os signos da interface são provenientes de diversos códigos subjacentes, caracterizando um hipercódigo. Tal

fato determina que se uma linguagem visual de interface viesse a ser projetada ela não abordaria todo o potencial expressivo que o medium interface pode oferecer.

Além disso, uma linguagem visual poderia restringir bastante a criatividade do designer e geraria interfaces com estruturas bastantes rígidas e significados bastante limitados e previsíveis. Embora se queira que em certos aspectos a interpretação seja restrita não se quer uma linguagem que restrinja a utilização de aplicações de maneira criativa e não prevista pelo designer. Nossa justificativa é equivalente às que alertam que o uso precoce de *linguagens de design* que acabam por limitar e enrijecer a usabilidade [Rheinfrank & Evenson 96].

UM SISTEMA SEMIÓTICO PARA APOIO AO DESIGN DE INTERFACES DE USUÁRIO

Vimos no capítulo anterior que um código é o instrumento de apoio a um processo comunicativo e, conseqüentemente, à produção da mensagem designer-usuário. De acordo com a teoria dos códigos de [Eco 76] e pelas características da interface como *medium* de metacomunicação através da qual a mensagem é transmitida, este código é, na verdade, um sistema semiótico formado por diversos outros códigos.

O Sistema Semiótico para Apoio ao Design de Interfaces de Usuário (SSADIU) auxilia o processo de design como produção de signos oferecendo um sistema de tipos-signos de interface que podem ser instanciados para a elaboração da mensagem do designer. Os tipos-signos correlacionam semanticamente tipos expressivos (widgets) a unidades de uma linguagem de descrição do modelo de usabilidade (os tipos semânticos).

O Processo de Design e o SSADIU

O nosso modelo teórico para a interface como expressão da mensagem possui um *medium* através do qual os *signos de interfaces* são veiculados e articulados em configurações espaciais e temporais. Os signos possuem características de interatividade e podem ser acionados pelos usuários através da *ferramenta de acionamento.* Os signos têm como expressão preferencial os widgets disponíveis na maioria das ferramentas de interfaces. Seus significados (objeto da mensagem) são os elementos do modelo de usabilidade.

O modelo de usabilidade como objeto da mensagem possui dois componentes. O modelo de funcionalidade descreve os signos do domínio e as funções aplicadas que os

manipulam. O modelo de interação deve descrever de maneira abstrata as estruturas de interações que compõem os *comandos* do sistema interativo e as *visualizações* de funções aplicadas e signos do domínio. Existem diversos métodos e linguagens para a especificação da funcionalidade e do modelo de interação de um sistema. Na abordagem da Engenharia Semiótica o designer precisa complementar esta especificação considerando que eles são parte do processo comunicativo existente entre o designer e o usuário. Este processo de especificação precisa ser complementado de maneira a permitir que o design da interface seja de fato a elaboração de uma mensagem, considerando todos os aspectos metacomunicativos.

O SSADIU deve oferecer um repertório de tipo-signos de interface. Cada tipo-SI do sistema associa um tipo expressivo a um tipo semântico. Os tipos expressivos são abstrações dos principais widgets disponíveis nas ferramentas de desenvolvimento de interfaces. Os tipos semânticos devem permitir descrever os elementos do modelo de usabilidade. No nosso sistema os elementos do modelo de usabilidade são descritos através de uma *linguagem de especificação (do modelo de usabilidade como objeto) da mensagem do designer.* As regras de mapeamento semântico associam sentenças desta linguagem a widgets.

Isto significa que a cada elemento do modelo de usabilidade que tenha sido concebido e especificado utilizando os tipos semânticos do sistema devem estar associados por regras de correlação semântica a um signo de interface. Signos do domínio, funções aplicadas, visualizações e, principalmente, comandos de funções devem poder ser descritos nesta linguagem para serem representados por widgets.

Na Engenharia Semiótica, o processo de design deve ser conduzido em duas etapas, como mostra a Figura 0-1. A primeira é a especificação do modelo de usabilidade utilizando a linguagem de especificação deste modelo onde é definida a semântica da mensagem do designer. A especificação semântica da interface é que define as ações que o usuário deve desempenhar sem preocupação com qual signo de interface elas devem estar associadas. O modelo semântico descrito pela linguagem especifica quais as ações a serem realizadas pelos usuários independente de quais widgets sejam utilizados.

Na segunda, com o apoio do sistema de tipos-signos é realizado um mapeamento entre as sentenças que descrevem o modelo de usabilidade e as instâncias dos signos de interface - os widgets de algum padrão de interface ou ferramentas de mercado. Após uma protótipação inicial ter sido feita, a sua avaliação deve ser realizada através de elaboração de cenários de uso, análise de tarefas com o protótipo e testes de usabilidade, para que se possa realizar os ajustes e se chegar a um protótipo final.

É importante ressaltar que o processo de design da usabilidade deve ser realizado apenas após as atividades da *análise de requisitos do domínio*. Estas atividades são: a *análise de perfis de usuários*, o *estabelecimento das necessidades (metas) dos usuários*, a *análise e modelagem de tarefas* e *análise funcional e de objetos*. Para apoiar estas atividades existem uma série de ferramentas e métodos dentro da Engenharia de Software, razão pela qual não nos demoraremos mais sobre o detalhamento e exploração da mesmas.

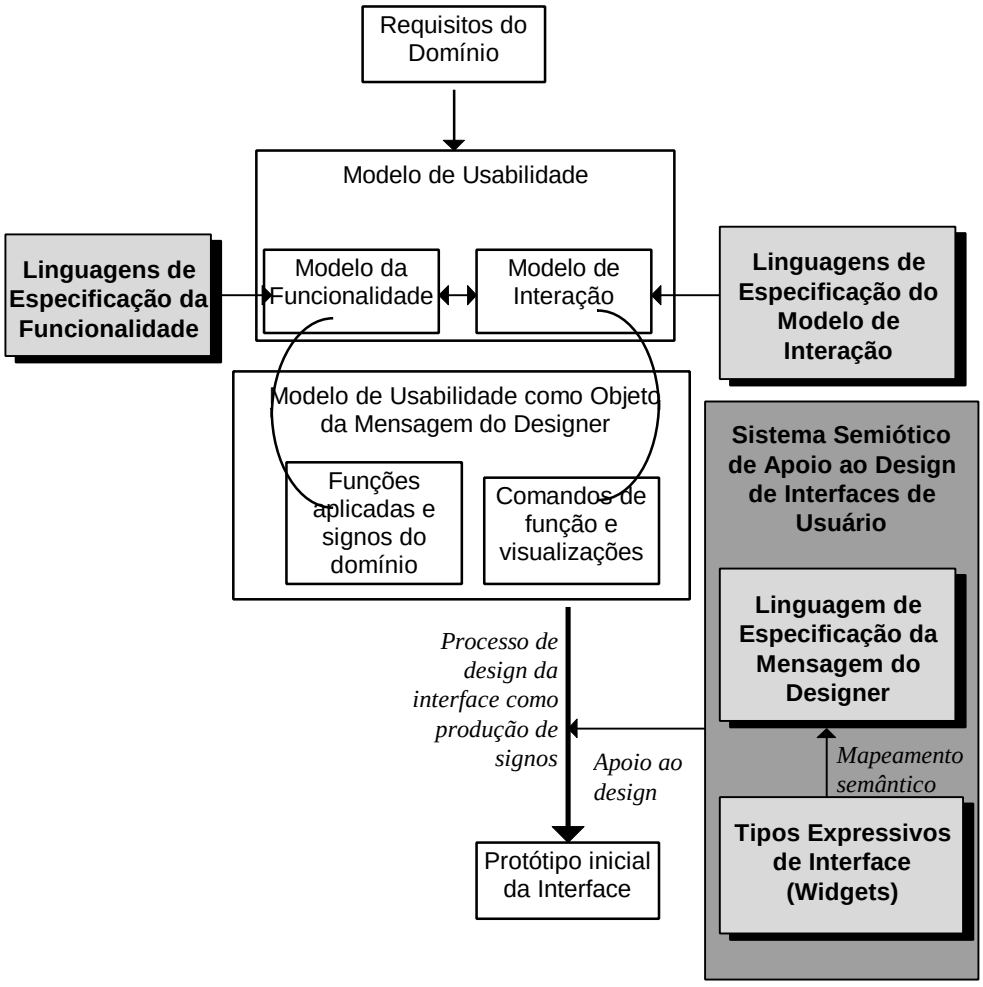


Figura 0-1: O processo de design e o SSADIU

A Linguagem de Especificação da Mensagem do Designer

A mensagem do designer deve ser utilizada para comunicar o modelo de usabilidade do sistema. A *Linguagem de Especificação da Mensagem do Designer* (*LEMD*) tem como objetivo apoiá-lo na formulação da mensagem sobre o modelo de usabilidade. Ela permite especificar os *signos do domínio, as funções aplicadas, as visualizações e os comandos* como sendo mensagens enviadas pelo design. As sentenças da linguagem descrevem as *mensagens de interação* que compõem a mensagem global, como discutido no capítulo 4. Estas mensagens são descritas de maneira abstrata independentes de quais signos de interfaces serão utilizados. Cada sentença da linguagem é uma instância de um tipo semântico do SSADIU e deve poder ser mapeada em *signos de interface* visando permitir a comunicação do modelo de usabilidade através da interface.

Uma linguagem de especificação complementar

Na engenharia de software a funcionalidade do sistema é determinada pelo processo de *especificação funcional de software.* Nesta especificação funcional o objetivo é determinar tudo o que o software deve fazer sem preocupação com a forma como o que é especificado será implementado, ou seja, com o seu funcionamento.

Nos métodos mais conhecidos da engenharia de software a funcionalidade é descrita através de um *modelo de especificação* utilizando alguma notação ou *linguagem de especificação.* Existem diversas notações para a especificação da funcionalidade. Elas podem ser *formais,* como VDM ou Z, ou *semi-formais* como *Diagramas de Fluxo de Dados (DFD), Diagrama de Estados, e Diagramas Entidade-Relacionamento* [Ghezzi 91]. Cada notação permite modelar certos aspectos específicos da funcionalidade. O DFD modela quais são os processos e recipientes de dados de um sistema sem determinar como estes processos podem ser controlados. Os diagramas de estados descrevem os eventos e estados associados a um determinado processador abstrato. As notações formais *baseadas-em-modelos,* como VDM e Z, modelam a funcionalidade utilizando dos conceitos matemáticos da teoria dos conjuntos através dos quais estruturas de dados são modelados e *funções aplicadas* são *mapeamentos* de estados iniciais em estados finais descritas em termos de relações lógicas. Cada linguagem de especificação, portanto, incorpora em sua semântica conceitos específicos que esquematizam a funcionalidade que será especificada.

A semântica das linguagens de programação descreve um programa como uma máquina dotada de funções, dados e recipientes de memória (variáveis). Dados são alocados em recipientes e funções podem ser realizadas com dados que estão nos recipientes. A semântica das linguagens de especificação permite modelar a funcionalidade em termos de *espaço de estados* que os dados podem ocupar quando transformados por funções aplicadas [Jones 86]. Ambas as linguagens oferecem limitações para a usabilidade do sistema como veremos em seguida.

A especificação da interação, também chamada de *design do diálogo,* requer a aplicação de formalismos específicos que permita estruturar as interações. Os formalismos baseados em redes de transição de estados, que permitem construir

estruturas de gramáticas de ações, devem ser utilizados. Também é necessário a aplicação de conhecimento sobre as capacidades e limitações cognitivas dos usuários.

A LEMD, por motivos argumentados no capítulo anterior, enfatiza a especificação dos componentes funcionais e de interação como elementos a serem comunicados ao usuário de maneira complementar à especificação funcional e da interação. A Figura 0-2 mostra o relacionamento do SSADIU com as linguagens de especificação funcional e da interação.

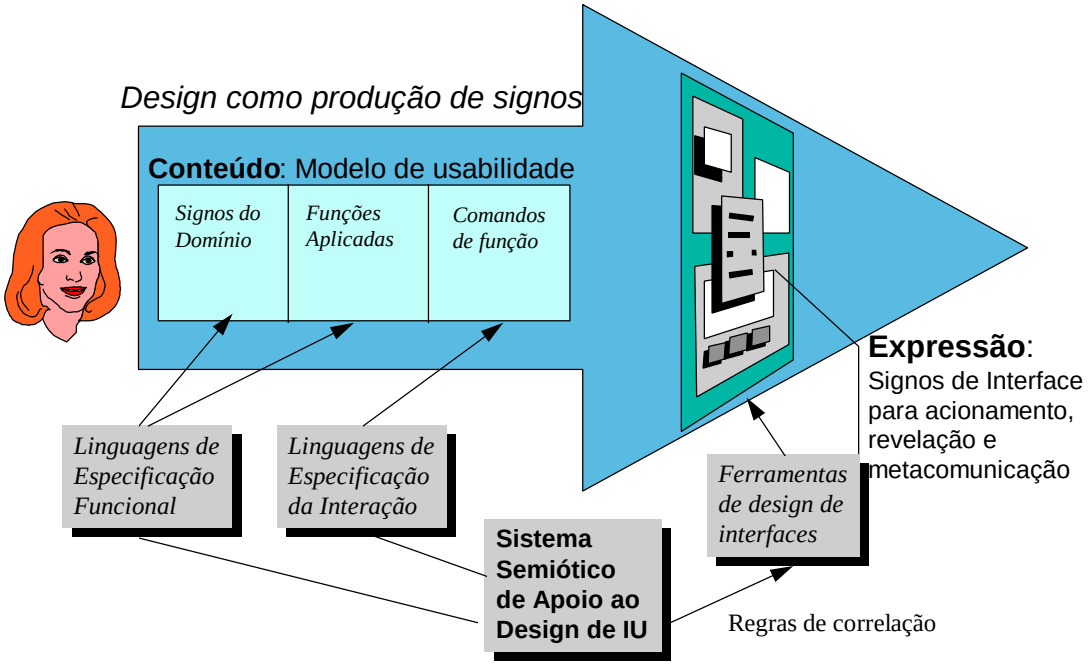


Figura 0-2: O SSADIU e as outras linguagens de especificação

Especificação de Signos do Domínio

Conforme discutido no capítulo anterior, a Engenharia Semiótica requer que a funcionalidade seja complementarmente especificada em termos de objetos que representam informações de conceitos do domínio. Visamos apenas descrever os signos do domínio que referencie adequadamente o conceito do domínio a ser representado e o tipo de representação da informação. A sentença que descreve signos do domínio possui a seguinte forma geral[21]:

[21]Os termos da linguagem são palavras do inglês visando uma ampla compreensão e aceitação nos meios de divulgação científica. A notação que utilizaremos para descrever a linguagem usa o fonte `Courier New`.

Domain-Sign *<domain-concept-name>* **representation-type** *<representation-type>*

A sentença abaixo descreve dois signos do domínio do tipo numérico:

Domain-Sign *Número da conta* **representation-type** number
Domain-Sign *Saldo* **representation-type** number

Especificação de Funções Aplicadas

As funções aplicadas devem ser descritas de tal forma que o designer possa associá-las com o comando que ele quer especificar. Apenas as funções do sistema, que podem ser controladas por comandos do usuário, devem ser especificadas. Na especificação devem ser descritos os operandos, que são objetos de representação de signos do domínio, e as condições anteriores e posteriores que determinam quais são as entradas e saídas desta função aplicada e devem ser utilizadas na especificação do comando.

A especificação de uma função aplicada é feita com a seguinte forma geral:

Application-Function *<Application-function-name>*
Operands *<Domain-signs-name>*
Pre-conditions *pre-conditions-list*
Post-conditions *post-conditions-list*
Control *<application-function-control>*
State *<application-function-state>*

A especificação das pré- e pós-condições devem ser descritas por uma linguagem de especificação com objetivos mais apropriados como a linguagem formal VDM [Jones 86]. A função aplicada *Consulta saldo*, por exemplo, pode ser especificada assim:

Application-Function *Consulta saldo*
Operands *Nome de cliente, Número da conta, Saldo*
Pre-conditions *pre-conditions-list*
Post-conditions *post-conditions-list*
Control *Iniciar, Cancelar*
State *Disponível, Executando, Terminado*

Um outra função aplicada *Calcula Lucro* pode ser especificada de maneira semelhante:

Application-Function *Calcula Lucro*
Operands *Valor de compra, Valor de venda, Lucro*

Pre-conditions *Valor de compra, Valor de venda* **must have an input value**
Post-conditions *Lucro = Valor de venda - Valor de compra*
Control *Start*
State *Disponível, Esperando-por-dado, Executando, Terminado*

Uma visão geral das mensagens

A estruturação do modelo de usabilidade realizada no capítulo anterior permitiu-nos identificar os componentes conceituais do modelo de usabilidade a serem comunicados ao usuário. Eles compreendem os *tipos semânticos* da mensagem do designer. De acordo como modelo metacomunicativo discutido no capítulo 4 a mensagem do designer é composta pelas mensagens de interação que são trocadas entre o usuário e sistema. Os tipos semânticos devem ser abstrações destas mensagens, permitindo especificá-las independentemene dos signos de interface que serão utilizados para expressá-las. A linguagem de especificação permite a descrição de maneira mais sistemática dos *tokens semânticos*: as sentenças descritas por ela.

Seguindo o modelo teórico descrito, a linguagem deve permitir ao designer especificar os signos do domínio, funções aplicadas, representações e, principalmente, comandos. Os comandos são especificados por estruturas de ações. Na abordagem da Engenharia Semiótica a mensagem do designer estende o modelo de usabilidade oferecendo o apoio à realização de tarefas através do envio de mensagens de metacomunicação direta e indiretamente através da interface. Com estes objetivos em mente e visando um mapeamento mais direto com os tipos expressivos, os tipos semânticos devem permitir a descrição do modelo de usabilidade sob a forma de mensagens a serem enviadas ao usuário. Estas mensagens compreendem:

Mensagens de *metacomunicação direta*, que permite ao designer enviar uma mensagem diretamente ao usuário para se referir a qualquer outro componente de usabilidade, inclusive à própria mensagem

Mensagens sobre *signos do domínio*, revelando o estado do sistema.

Mensagens sobre *funções da aplicação*, revelando o seu estado operacional.

Mensagens sobre *interações básicas* indicando ao usuário a interação a ser desempenhada.

Mensagens sobre a *estrutura sintática dos comandos*, ou seja, a estrutura e a articulação das interações que o usuário precisa desempenhar.

Mensagens de *metacomunicação para apresentação e controle da leitura da mensagem* que comunicam como o usuário deve ler a própria mensagem do designer. Esta mensagem pode ser veiculada tanto pelos signos de acionamento quanto pelos de revelação.

Vamos a seguir mostrar como a linguagem permite ao designer construir as sentenças que especificam o modelo de usabilidade como mensagens que ele deve enviar

ao usuário para cumprir seu propósito metacomunicativo O designer vai ser mais bem sucedido dependendo da maneira como ele estruturar os componentes do modelo de usabilidade na sua mensagem e da escolha dos signos de interface mais adequados para expressá-los. A Figura 0-3 mostra cada uma destas mensagens integrando as mensagem global do designer.

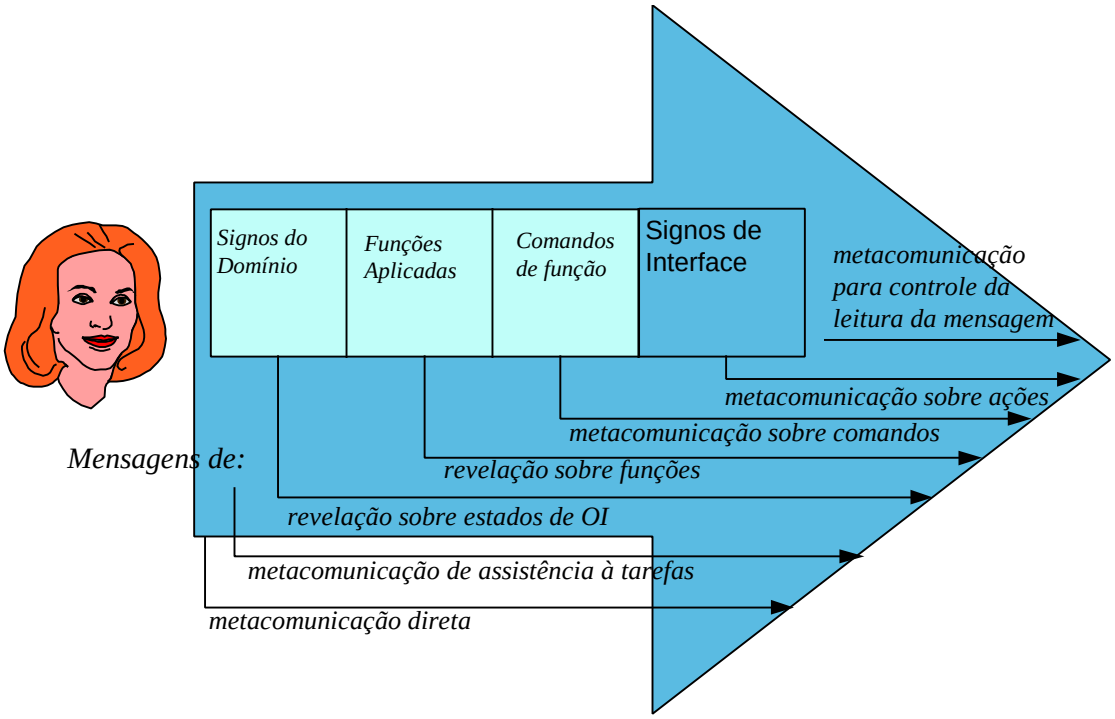


Figura 0-3: As mensagens do designer

Mensagem de Metacomunicação Direta

As mensagens diretas são metacomunicações explícitas que o designer envia para o usuário. Elas se diferenciam das indiretas que realizam uma comunicação através do processo de acionamento ou de revelação do próprio signo de interface.

Por exemplo, para o designer enviar uma mensagem direta para o usuário ele se utilizaria da sentença:

```
View "Hello User!"
```

Esta sentença da linguagem indica que o designer deseja enviar a mensagem "Hello User!" para o usuário. Esta mensagem deve ser enviada por algum signo de interface como pode ser o caso uma *caixa de mensagem* (*message box*). Na especificação

abstrata o designer não deve se preocupar com os signos de interface nem com o layout da sua mensagem.

Mensagens de Signos do Domínio e Funções Aplicadas

A mensagem que o designer envia é formada por diversas outras mensagens que são apresentadas durante o processo de interação. Os *signos do domínio* são informações que precisam ser mostradas para o usuário com o objetivo de ajudá-lo a formular a meta ou avaliar a execução de um comando. Eles são mostrados através de *objetos de representação.* O objeto de representação, quando apresentado para o usuário, é uma mensagem que diz "veja a informação do domínio".

A especificação desta mensagem é feita da seguinte forma:

```
View Information-of <domain-sign-name>
```

Para o exemplo de transações bancárias, o designer pode especificar:

```
View Information-of Nome de Cliente
View Information-of Extrato
```

Da mesma forma, a existência de uma função aplicada bem como o seu estado devem ser comunicadas ao usuário. A função aplicada é comunicada junto com o comando que permite controlar a sua execução. Os seus diversos estados são comunicados com mensagens do tipo "*veja o estado da função*". Os estados de uma função aplicada podem ser *executando, terminada, suspensa, disponível, não-disponível, em-espera.*

A especificação da mensagem de visualização de um estado da função aplicada possui a seguinte forma geral:

```
View State-of <application-function-name>
```

O *estado da função aplicada* é também uma informação de tempo de execução. Por exemplo, a função *Consulta saldo* pode estar nos estados *executando* ou *terminado.* Na especificação esta informação pode ser omitida, deixando para o software da interface o papel de implementá-la.

```
View State-of Consulta saldo
```

É importante ressaltar que a *visualização* é um tipo de interação básica. Isto enfatiza o aspecto metacomunicativo da Engenharia Semiótica no qual a mensagem global do designer é veiculada por mensagens de interação.

Mensagens de Interação Básica

O conceito de *interação básica* refere-se às operações perceptivas, motoras e cognitivas realizadas pelo usuário diretamente na interface. Elas podem ser descritas de maneira abstrata de acordo com os seguintes tipos abstratos: *acionamento, fornecimento de caracteres, seleção e visualização*. A visualização é utilizada para a revelação de signos do domínio e dos estados de funções aplicadas como visto na sub-seção anterior. As outras são utilizadas para controle de funções, de mensagens, fornecimento e seleção de informações de signos do domínio.

As formas gerais das mensagens de interação para cada um dos casos é:

```
Enter Information-of <domain-sign-name>
Select Information-of <domain-sign-name>
Activate    <application-function-control>    Application-Function
<application-function-name>
Activate <message-control> Command-Message <command-name>
```

Para que o usuário forneça o *nome do cliente, número da conta e o valor creditado* o designer pode enviar as seguintes mensagens para ele.

```
Enter Information-of Nome de cliente
Enter Information-of Número da conta
Enter Information-of Valor creditado
```

A seleção deve ser utilizada para o fornecimento de informações para os tipos de representação cujas instâncias formam um conjunto limitado e pré-definido, como por exemplo, os produtos de venda de uma empresa:

```
Select Information-of Nome-do-produto
```

Os números de conta de um mesmo cliente poderiam ser selecionados de uma lista após o cliente ter fornecido o seu nome:

```
Enter Information-of Nome de cliente
Select Information-of Número da conta
```

A interação de acionamento é usada para ativação de controles de funções aplicadas e para controle de leitura da mensagem. O controle de leitura será discutido mais adiante na sub-seção 0.

Para a função aplicada *Consulta saldo*, o designer deve comunicar os controles operacionais *iniciar* e *cancelar*:

```
Activate Start Application-Function Consulta Saldo
Activate Waive Application-Function Consulta Saldo
```

Mensagens de Comandos de Função

Os comandos de função possuem uma sintaxe e uma semântica. A sintaxe do comando é uma estrutura de articulação de interações básicas e podem ser dos seguintes tipos de articulação: *seqüência, agrupamento, combinação, repetição e seleção*. Para facilitar a etapa de *especificação da estrutura de ações* o designer deve comunicar através da aparência, do comportamento e de configurações espaciais e temporais de signos de interfaces, qual a estrutura do comando que o usuário pode executar.

Estes tipos de articulação estão de acordo com os padrões de redes de transição de estados discutidos no capítulo anterior. Isto significa que especificações de modelos de interação realizados por formalismos de redes de transição ou de gramáticas de ação podem ser mapeados na nossa linguagem e especificados como mensagens do designer.

O designer deve especificar uma mensagem sobre a estrutura de comandos indicando qual o tipo de estrutura e quais as interações básicas componentes, ou, recursivamente, uma outra estrutura. Na especificação da mensagem, o designer pode utilizar uma especificação de interação feita via um formalismo específico e mapear nos tipos da nossa semântica.

As formas gerais para cada um os tipos de estruturas de articulação é:

```
Sequence { <interaction-messages> }
Select { <interaction-messages> }
Join { <interaction-messages> }
Repeat { <interaction-messages> }
Combine { <interaction-messages> }
```

Uma <*interaction-message*> é uma das mensagens de interação básicas descritas anteriormente. Vejamos como exemplo, uma estrutura de interações associada com

```
Sequence {
   Enter Information-of Valor de compra
   Enter Information-of Valor de venda
   Activate Start Application-Function Calcula Lucro
   View Information-of Lucro
}
```

Caso as informações não precisem ser fornecidas numa ordem explícita elas podem ser especificadas com o *agrupamento* (*Join*)

```
Join {
   Enter Information-of Valor de compra
   Enter Information-of Valor de venda
}
```

A seleção (*Select*) pode ser utilizada, por exemplo, para indicar ao usuário que ele pode escolher o controle operacional que ele deseja acionar:

```
Select {
   Activate Start Application-Function Consulta saldo
   Activate Waive Application-Function Consulta saldo
}
```

A mensagem *repetição* (*Repeat*) pode ser utilizada quando uma interação deve ser realizada de maneira repetitiva. O número de repetições é controlado pelo software e pode ser indeterminado ou pré-determinado. Um caso de interação repetitiva é o fornecimento de diversas informações de um mesmo signo do domínio como, por exemplo, os diversos nomes de cliente a serem cadastrados. A função aplicada deve determinar se o número de clientes é pré-determinado ou se o usuário deve fornecer repetidamente até que, quando fornecer um certo valor, a repetição é encerrada.

```
Repeat {
   Enter Information-of Nome de cliente
}
```

A mensagem *combinação* (*Combine*) é utilizada quando o usuário deve realizar duas interações simultâneas ou dependentes. A interação é combinada quando é dependente da ocorrência de outra, como, por exemplo, o caso de selecionar dois signos do domínio que dependem um do outro. Esta mensagem difere-se da seqüência, pois é

necessário comunicar para o usuário a dependência que existe entre os dois que é de fundamental importância para a compreensão da funcionalidade

```
Combine {
   Select Information-of Nome do Arquivo
   Select Information-of Diretório
}
```

A combinação ocorre também quando o usuário combina uma tecla de controle como *shift, ctrt, ou alt* com outra tecla ou com o acionamento do mouse. Entretanto, neste caso a especificação é dependente do widget ou do dispositivo de interação e não está no nível de interação abstrata que desejamos.

A estruturas de interações podem ser **aninhadas** recursivamente.

```
Sequence { <interaction-structure> }
Repeat { <interaction-structure> }
Join { <interaction-structure> }
Combine { <interaction-structure> }
Select { <interaction-structure> }
```

Onde <*interaction-structure*> é um conjunto de mensagens de interação ou uma outra estrutura. Por exemplo, juntando os casos exemplificados, podemos estruturá-los todos juntos, da seguinte forma:

```
Sequence {
    Join {
       Enter Information-of Nome de Cliente
       Enter Information-of Número da Conta
    }
    Select {
       Activate Start Application-Function Consulta saldo
       Activate Waive Application-Function Consulta saldo
    }
   View Information-of Saldo
}
```

A semântica do comando é determinada pela função aplicada e pelo tipo de controle operacional associado a ela. Tanto a função aplicada quanto o tipo do controle são comunicados ao usuário através de seus nomes.

Para uma melhor estruturação e controle do designer de forma a possibilitar que as mensagens de um mesmo comando possam ser apresentadas diversas vezes sem ter

que repeti-las, faz-se necessário introduzir o conceito de *mensagem de comando.* Uma mensagem de comando estrutura todas as ações necessárias para a ativação de uma função aplicada e dá-lhe um nome para referência em sua reutilização.

A construção para a especificação de uma mensagem de comando é:

```
Command-Message <command-name> for Application-Function
<application-function-name>
```

Por exemplo, podemos introduzir a mensagem de comando para a função aplicada *Consulta Saldo* definda acima:

```
Command-Message Consultar Saldo for Application-Function Consulta
Saldo
      Sequence {
          Join {
            Enter Information-of Nome de Cliente
            Enter Information-of Número da Conta
          }
          Select {
            Activate Start Application-Function Consulta saldo
            Activate Waive Application-Function Consulta saldo
          }
          View Information-of Saldo
      }
```

Vamos mostrar como as estruturas e mensagens apresentadas até agora descreveriam um exemplo real de interface. Vejamos na figura 0-4 a caixa de diálogo para o comando *abrir arquivo*, que é bastante comum na maioria das interfaces de aplicativos do ambiente MS Windows.

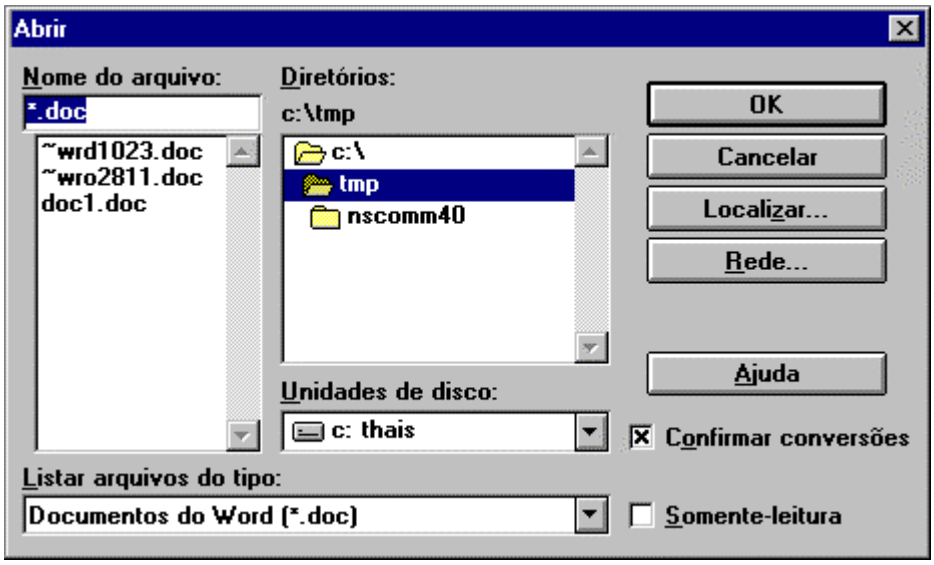


Figura 0-4: Exemplo de Mensagem de Comando

```
Command-Message Abrir for Application-Function Abrir Arquivo
      Join {
            Combine {
                  Select {
                        Enter Information-of Nome do Arquivo
                        Select Information-of Nome do Arquivo
                  }
                  Select Information-of Diretorios
                  Select Information-of Unidades de disco
                  Select Information-of Listar arquivos do tipo
                  Enter  Information-of Confirmar conversões
                  Enter  Information-of Somente-leitura
            }
            Select {
                  Activate Show Command-Message Localizar
                  Activate Show Command-Message Rede
            }
            Activate Start Application-Function Abrir
            Activate Waive Application-Function Abrir
      }
```

A mensagem **Activate** <message-control> **Command-message** indica que o usuário pode, a partir do processo de comando, acionar um outro comando. No caso acima é o comando *Localizar* ou comando *Rede*. As mensagens de ativação dizem respeito à coisas conceitualmente distintas que serão ativadas. Nesta mensagem, uma outra mensagem de comando é mostrada para que um novo comando possa ser realizado. No exemplo, esta distinção é comunicada ao usuário pelo símbolo "**...**" logo ao lado do rótulo.

Mensagens de Tarefas

Vimos que uma meta do usuário é satisfeita pelo planejamento de tarefas compostas por vários comandos. A *distância semântica* no mapeamento tarefa-ação corresponde ao esforço de planejamento quando para um deteminada meta do usuário não existe uma função aplicada no sistema. O planejamento corresponde à atividade mental de dividir sua meta incial em sub-metas que possam ser satisfeitas com as funções aplicadas existentes.

Com o objetivo de diminuir a *distância semântica* quando as funções aplicadas e os seus respectivos comandos estão num nível mais baixo da árvore, o designer pode elaborar ***mensagens de tarefas*** que funcionam como *assistentes* (*wizards*) e orientam o usuário sobre como executar um conjunto de comandos necessários para uma determinada meta de mais alto nível. As mensagens de tarefas visam orientar o usuário no planejamento e execução dos planos. O designer pode apresentar as mensagens com comandos necessários para se atingir uma meta e a maneira com eles devem ser articulados. As *mensagens de tarefas* comunicam grupos de comandos e podem ser consideradas extensões de mensagens de comandos. A diferença é que cada comando está associado à uma função aplicada enquanto uma mensagem de tarefas está associada a diversas funções aplicadas, um vez que ela é composta por vários comandos de função.

Os grupos de comandos podem ser articulados de maneira semelhante às estruturas de articulação de interações básicas, ou seja, comandos podem ser agrupados como *seqüências, agrupamentos, seleções, repetições e combinaçã*o. A *seqüência* indica uma ordem seqüência linear de execução dos comandos. No *agrupamento* a ordem na qual os comandos são executados é aleatória. A *seleção* indica que o usuário pode tomar decisões sobre qual comando executar. A *repetição* é usada quando um mesmo comando precisa ser executado várias vezes para que uma meta seja atingida. Finalmente, a *combinação* de comandos ocorre quando eles precisam ser ativados conjuntamente para que as funções aplicadas correspondentes sejam executadas de maneira síncrona ou simultânea.

A articulação de comandos em mensagens de tarefas permitem a realização de planos que satisfazem ao formalismo *GOMS (Goals, Operators, Methods and Selection*

Rules) [Card, Moran & Newell 83][22]. Este formalimo permite elaborar modelos analíticos das tarefas que o usuário realiza e contêm apenas as articulações *seqüência* e *seleção.* Como o objetivo do GOMS é analítico, a descrição de um *método* corresponde a seqüência na qual o usuário realmente desempenhou suas tarefas, obedecendo ou não estruturas de linearidade, repetição, ou agrupamento. [Hix & Hartson 93] apresentam outros tipos de articulação de tarefas que não podem ser especificados pela nossa linguagem.

A forma geral para a especificação de uma mensagem de tarefa é:

```
Task-Message <Task-name>
  <structure> { <command-name> }
```

Vejamos como uma mensagem de tarefa pode comunicar ao usuário que uma meta deve ser atingida pela execução de outros comandos. Vamos supor que o usuário deseja realizar uma transferência de dinheiro e que o sistema de transação bancária não possua a função aplicada *transferência.* O designer pode auxiliá-lo comunicando uma mensagem de tarefa que comunique que uma transferência deve ser realizada pela articulação de três outros comandos,

```
Command-Message Saque for Application-Function Saque
      Sequence {
        Enter Information-of Nome do Cliente
        Enter Information-of Valor Debitado
        Activate Start Application-Function Saque
      }

Command-Message Depositar for Application-Function Depósito
      Sequence {
        Enter Information-of Nome do Cliente
        Enter Information-of Valor Creditado
        Activate Start Application-Function Depósito
      }

Task-Message Transferência
      Sequence {
            Command-Message Sacar
            Command-Message Depositar
      }
```

[22]Ver capítulo 3, seção Abordagens Cognitivas.

Mensagens de Controle

As mensagens discutidas até agora, em sua maioria, dizem respeito diretamente ao desempenho do usuário no planejamento e execução das tarefas, comandos e ações, conforme nosso modelo. Vimos como mensagens de comandos são descritas através da *articulação de interações* e como tarefas podem ser desempenhadas com auxílio de *mensagens de tarefas.*

Um sistema pode ter dezenas ou centenas de comandos cujas mensagens precisam ser comunicadas aos usuários de maneira clara e organizada. É preciso acrescentar na linguagem recursos para a especificação das mensagens que apresentam as outras mensagens (visualização de signos e estado de funções, interações básicas e comandos).

O conceito de *controle de leitura de mensagens* permite ao usuário ativar e desativar mensagens de comandos ou de tarefas de acordo com as suas necessidades, controlando o processo de leitura das outras mensagens que formam a mensagem do designer. Este conceito revela-nos uma importante distinção entre o *acionamento de controles de comandos* - que provoca mudanças nos elementos da funcionalidade do sistema - e o *acionamento dos controles de leitura* - que provoca mudanças apenas na interface de usuário. Este é o caso das maioria dos botões de comando cujo rótulo contém o símbolo três pontos (“**rótulo...**”). Outros signos de interfaces mais elucidativos podem ser utilizados.

O controle de leitura deve ser realizado por uma *interação básica* do tipo *acionamento.* Esta interação básica é comunicada ao usuário por uma *mensagem de interação,* como indicado anteriormente. Os tipos básicos de controles de leitura <message-control> são *mostrar, fechar, maximizar e minimizar.*

A forma desta mensagem é

```
Activate <message-control> Task-Message <task-name>
Activate <message-control> Command-Message <command-name>
```

Os controles de leitura podem fazer parte de *mensagens de comandos* e de *mensagens de tarefas.* O exemplo do comando *Abrir* visto na sub-seção 0, inclui

mensagens de acionamento para *controle de comando de função aplicada* e *controle de mensagens de comando.*

Os controles de leitura podem ser agrupados em *mensagens de controle* que, por sua vez, podem agrupar controles de *mensagens de comandos, de tarefas* e, também *mensagens de controle.* Os *controles de funções aplicadas* também podem ser incluídos numa mensagem de controle. A forma geral de uma *mensagem de controle* é:

```
Control-Message <control-name>
```

Tal como as *mensagens de comandos* e de *tarefas*, as *mensagens de controle* também permitem a especificação de estruturas de articulação de controles de leitura usando os mesmos controles: *seqüência, agrupamento, repetição, combinação* e *seleção.*

Com estas estruturas o designer pode especificar mensagens para o usuário, ativar as mensagens de comandos e, então, comandar as funções aplicadas. No exemplo abaixo mostramos duas mensagens de controle, uma com a estrutura de articulação *seqüência* e a outra com a *seleção* para os mesmo controles. Vamos utilizar as mensagens de comando *Saque* e *Depósito* especificada em exemplos anteriores.

```
Control-Message Transferência
      Sequence {
            Activate Start Command-Message Saque
            Activate Start Command-Message Depósito
      }

Control-Message Selecione-Transação
      Select {
            Activate Start Command-Message Saque
            Activate Start Command-Message Depósito
      }
```

Nestes exemplos, mensagens correspondentes a controles para a ativação das mensagens de comando são apresentadas ao usuário. No primeiro caso eles são apresentados em seqüência, e no segundo, o mais comum, eles são oferecidos para que o usuário escolha o que quer comandar. Na mensagem de controle é apresentada uma mensagem que comunica que o usuário pode ativar o comando *Saque* e, em seguida, a mensagem sobre o acionamento do comando *Depósito.*

É importante ressaltar a diferença entre o "**Activate Command-Message** *nome-da-mensagem*" e "**Command** *nome-da-mensagem*" que ocorre como parte das *mensagens de tarefas*. Neste primeiro caso o designer está comunicando um *signo de interface* que permite ao usuário acionar a *mensagem de comando*, enquanto que no segundo a *mensagem de comando* é apresentada diretamente por decisão do designer.

Aplicação de mensagens de controles: *Menus*

Um sistema interativo pode oferecer a um usuário centenas de funções aplicadas. Os comandos para estas funções aplicadas não podem ser todos comunicados ao mesmo tempo para o usuário. Um dos motivos é a limitação do medium interface. Outro é a capacidade do usuário de encontrá-los.

A principal utilização da estrutura de mensagens de controle é para a seleção de comandos de funções aplicadas. O comandos de funções aplicadas são apresentadas ao usuário em estruturas de *menus de controles acionadores*. Os menus contêm diversas funções aplicadas disponíveis para escolha e conseqüente ativação da mensagem de comando correspondente.

Existem diversas estruturas de menus. Um *menu linear* é uma lista de opções de mensagens de controles acionadores que permite ao usuário escolher e acionar uma mensagem de comando correspondente.

```
Control-Message Menu-correio
      Select {
            Activate Show Command-Message Ler
            Activate Show Command-Message Enviar
            Activate Show Command-Message Arquivar
            Activate Show Command-Message Imprimir
      }
```

Os *menus hierárquicos* contêm diversas listas de controles aninhadas numa árvore cujas folhas contêm os controles acionadores de comandos. Nos nós da árvore que não são folhas, controles acionadores de *mensagens de controle* devem ser especificados. Um menu hierárquico possui a seguinte especificação

```
Control-Message Menu-Editor
      Select
            Activate Show Control-Message Menu-Arquivo
            Activate Show Control-Message Menu-Editar
      }
```

```
Control Message Menu-Arquivo
      Select {
            Activate Show Command-Message Novo
            Activate Show Command-Message Abrir
            Activate Start Application-function Fechar
            Activate Start Application-function Salvar
            Activate Show Command-Message Salvar-como
      }

Control Message Menu-Editar
      Select {
            Activate Start Application-function Cortar
            Activate Start Application-function Copiar
            Activate Start Application-function Colar
      }
```

Aplicação de mensagens de tarefas: *Assistentes*

As *mensagem de tarefas* usadas em associação com *controles de leitura* permitem a construção de assistentes de tarefas (wizards). Ao contrário das mensagens de comandos cujo objetivo é comunicar uma estrutura rígida (necessária e suficiente) de interações para comandar funções aplicadas, o objetivo das mensagens de tarefas é principalmente motivar e orientar o usuário. Associadas aos controles de leitura, as mensagens de tarefas devem poder ser controladas pelos usuários permitindo uma maior flexibilização em relação aos comandos que o usuário quer realizar. Vejamos um *template* da especificação de um assistente.

```
Task-Message Message1
      Select {
            Command command1
            Activate Show Task-Message message2
      }
Task-Message Message2
      Select {
            Command command2
            Activate Show Task-Message message1
            Activate Show Task-Message message3
      }
Task-Message Message3
      Select {
            Command command3
            Activate Show Task-Message message2
            Activate Show Task-Message message4
      }
Task-Message Message4
      Select {
            Command command4
            Activate Show Task-Message message3
      }
```

A mensagens acima especificam um modelo de interação no qual o designer deve realizar as tarefas numa estrutura seqüencial interativa. O usuário realiza o comando1 quando a respectiva mensagem lhe é mostrada junto com o controle para mostrar a *mensagem de tarefa message*-2. Nesta última ele tem a opção de escolher entre executar o *comando2*, voltar para a *tarefa1* ou ir para a *tarefa3*. Nesta especificação o designer optou por não utilizar a estrutura de tarefas "**Sequence**" de maneira a oferecer liberdade ao usuário de controlar a execução da seqüência, permitindo inclusive a *volta-atrás.*

Outro exemplo mais completo:

```
Task-Message Main-task
      Select {
            Activate Show Task-Message Sub-task1
            Activate Show Task-Message Sub-goal2
      }
Task-Message Sub-task1
      Sequence {
            Command command1
            Command command2
            Command command3
            Activate Show Task-Message Main-task
      }
Task-Message Sub-task2
      Select {
            Sequence {
                  Command command1
                  Command command2
                  Command command3
            }
            Activate Show Task-Message Main-task
      }
```

A primeira mensagem apresenta para o usuário um menu para a ativação de mensagens de tarefas que pode permitir que ele desempenhe as tarefas *Sub-task1* e *Sub-task2*. A mensagem para *Sub-task1* comunica uma seqüência de mensagens de comandos e, no final, comunica um signo que permite a ativação da mensagem de tarefa principal. Na mensagem *Sub-task2* o usuário pode selecionar entre realizar a seqüência de comandos ou voltar para a mensagem principal.

A nossa linguagem pressupõe que cada vez que uma mensagem de comando, de tarefa ou de controle é ativada, a mensagem anterior é imediatamente desativada. Assim, quando o designer quiser esperar que o usuário controle esta desativação, ele deve utilizar

um controle de mensagem comunicando ao usuário que ele deve acionar o controle *fechar*.

Sintaxe da Linguagem de Especificação

Tendo descrito a semântica informal da linguagem de especificação, descreveremos aqui a sua sintaxe.

Gramática

A gramática que descreve as sentenças da linguagem pode ser definida utilizando a notação BNF. As convenções utilizadas nesta notação são descritas na Tabela 0-1.

Convenções:

< >	indicam símbolos não-terminais da linguagem
' '	indicam símbolos terminais da linguagem
[ ]	indicam UMA ou mais ocorrências do(s) elemento(s) delimitado(s)
{ }	indicam ZERO ou mais ocorrências do(s) elemento(s) delimitado(s)
( )	indicam um grupamento do(s) elemento(s) delimitado(s)
::=	indicam uma produção gramatical
|	indica uma produção alternativa

Tabela 0-1: Convenções da notação BNF que descreve a gramática.

Descrição da Mensagem Global de Interface

<interface-message>::= [<task-message> | <command-message> | <control-message>]

Descrição das Mensagens

<command-message> ::= <command-message-header> <command-message-body>

*<command-message-header> ::= **'Command-Message'** <message-name> **'for Application-Function'** <application-function-name>*

*<command-message-body> ::= <interaction-message> | <structure> **'{'** [<command-message-body>] **'}'***

<interaction-message> ::= <activation-sentence> | <information-sentence> | <visualization-sentence> | <metacomm-sentence>

<task-message> ::= <task-message-header> <task-message-body>

*<task-message-header> ::= **'Task-Message'** <message-name>*

*<task-message-body> ::= <structure> **'{'** [<command-sentence> | <activation-sentence> | <visualization-sentence> | <metacomm-sentence>]**'}'***

<control-message> ::= <control-message-header> <control-message-body>

<control-message-header> ::= ***'Control-Message'*** *<message-name>*

*<control-message-body> ::= <structure> '**{**' [<activation-sentence> | <metacomm-sentence>]'**}**'*

<structure>::= ***'select'*** **|** ***'repeat'*** **|** ***'sequence'*** **|** ***'join'*** **|** ***'combine'***

Descrição de Sentenças

<activation-sentence>::= ***'Activate'*** *<application-function-control-sentence> | <message-control-sentence>*

<application-function-control-sentence> ::= <application-function-control> ***'Application-Function'*** *<application-function-name>*

<application-function-control> ::= ***'Start'*** **|** ***'Stop'*** **|** ***'Suspend'*** **|** ***'Continue'*** **|** ***'Step-through'*** **|** ***'Waive'*** **|** ***'Undo'***

*<message-control-sentence> ::= <message-control> (**'Command-Message'**|**'Task-Message'**|**'Command-Message'**) <message-name>*

<message-control> ::= ***'Show'*** **|** ***'Discard'*** **|** ***'Minimize'*** **|** ***'Maximize'***

<command-sentence> ::= ***'Command'*** *<command-message-name>*

*<information-sentence> ::= (**'Enter'**|**'Select'**)* ***'Information-of'*** *<domain-sign-name>*

<visualization-sentence> ::= ***'View'*** *(**'Information-of'** <domain-sign-name> |* ***'State-of'*** *<application-function-name>)*

<metacomm-sentence>::= ***'View'*** *<direct-message>*

<direct-message>::= ANY-TEXT

Especificação de Funções Aplicadas

<application-function> ::= ***'Application-function'*** *<application-function-name>* ***'Operands'*** *[<operand>]* ***'Pre-conditions'*** *<pre-c>* ***'Post-conditions'*** *<post-c>* ***'Control'*** *<application-function-control>* ***'States'*** *<application-function-state>*

<operands> ::= <domain-sign>

<pre-c>::= pre-conditions-list

<post-c> ::= post-conditions-list

<*application-function-state*>::= **'available' | 'unavailable' | 'running' | 'finished' | 'suspended' | 'waiting-for-input'**

Especificação de Signos do Domínio

<*domain-sign*> ::= **'Domain-Sign'** <*domain-sign-name*> **'Representation-type'** <*representation-type*>

<*representation-type*> ::= <*simple-rep-type*> | <*structured-rep-type*>

<*simple-rep-type*> ::= **'number' | 'text' | 'truth-value'**

<*structured-rep-type*> ::= {**'set-of' | 'hierarchy-of' | 'network-of' | 'table-of'**} <*simple-rep-type*>

Nomes

<*domain-sign-name*> ::= ANY-CHOSEN-NAME

<*message-name*> ::= ANY-CHOSEN-NAME

<*application-function-name*> ::= ANY-CHOSEN-NAME

Vocabulário

Os termos da linguagem e uma descrição informal do significado e da aplicação de cada um deles estão descritos na Tabela 0-2.

Activate	Interação básica abstrata na qual o usuário deve desempenhar um acionamento simples com auxílio ou não de ferramenta de acionamento.
Application-Function	Função aplicada que modifica um estado do sistema de acordo com a sua especificação.
Available	Estado de função disponível para executar de uma função aplicada.
Combine	Estrutura de articulação que indica simultaneidade e dependência.
Command	Comando mostrado ao usuário por iniciativa do designer.
Command-Message	Especifição de uma mensagem de comando.
Continue	Tipo de controle operacional de uma função aplicada para continuação de uma operação suspensa.
Control	Especificação de controle operacional de uma função aplicada.
Control-Message	Especifição de uma mensagem de controle.
Discard	Tipo de controle de uma mensagem que faz como que ela seja “fechada”.

Domain-Sign	Signo do domínio que representa uma informação a respeito de um conceito do domínio.
Enter	Interação básica abstrata na qual o usuário deve fornecer caracteres alfanuméricos.
Finished	Estado de "terminada" de uma função aplicada.
for Application-Function	Termo que associa um comando com uma função aplicada.
Hierarchy-of	Tipo de representação estruturada - em hieraquia.
Information-of	Especifica que uma informação do domínio deve ser visualizada.
Join	Estrutura de articulação na qual a ordem é aleatória.
Maximize	Tipo de controle de uma mensagem que faz como que ela seja maximizada.
Minimize	Tipo de controle de uma mensagem que faz como que ela seja minimizada.
Network-of	Tipo de representação estruturada - em rede.
Number	Tipo de representação simples - numérica.
Operands	Especificação dos operandos de uma função aplicada.
Pre-condition	Especificação das pré-condições de uma função aplicada.
Post-condition	Especificação das pós-condições de uma função aplicada.
Repeat	Estrutura de articulação que indica repetição.
Representation-type	Especificação do tipo de representação.
Running	Estado de "Executando" de uma função aplicada.
Select	Estrutura de articulação que indica seleção de interações ou seleção de uma informação de um conjunto limitado do tipo de representação.
Sequence	Estrutura de articulação que indica seqüência.
Set-of	Tipo de representação estruturada - conjunto.
Show	Tipo de controle de uma mensagem que faz como que ela seja mostrada.
Start	Tipo de controle operacional de uma função aplicada para iniciar uma função aplicada disponível.
State-of	Especifica que o estado de uma função aplicada deve ser visualizado.
Step-through	Tipo de controle operacional de uma função aplicada para executar passo-a-passo uma função aplicada disponível.
Stop	Tipo de controle operacional de uma função aplicada para finalizar uma função aplicada disponível.
Suspend	Tipo de controle operacional de uma função aplicada para

	suspender uma função aplicada disponível.
Suspended	Estado de "Suspenso" de uma função aplicada.
Table-of	Tipo de representação estruturada - tabela.
Task-Message	Especificação de uma mensagem de tarefa
Text	Tipo de representação simples - texto.
Truth-value	Tipo de representação simples - valor-verdade.
Unavailable	Estado de "Ocupado" de uma função aplicada.
Undo	Tipo de controle operacional de uma função aplicada para desfazer uma função executada.
View	Interação básica abstrata na qual o usuário deve visualizar algo.
Waiting-for-input	Estado de "Esperando-por-dado-de-entrada" de uma função aplicada.
Waive	Tipo de controle operacional para cancelar (desitir de iniciar) uma função aplicada.

Tabela 0-2: O vocabulário da Linguagem de Especificação

Os Tipos-Signos de Interface

Os elementos do modelo de usabilidade devem ser comunicados ao usuário através do tipos-signos de interface do SSADIU. Nas seções anteriores vimos os tipos semânticos que permitem especificar o modelo de usabilidade na forma de mensagens a serem comunicadas ao usuário.

A restrição de nossa abordagem ao design de interface GUI/WIMP permite-nos utilizar a noção de widget como forma de permitir a viabilização computacional do conceitos de signo de interface.

Nesta seção vamos mostrar como os widgets podem ser vistos como tipos expressivos associados aos tipos semânticos. Vamos apresentar um repertório de tipo-signos descrevendo os *significados preferenciais* de cada widget fazendo um mapeamento dos mais conhecidos com os tipos semânticos descritos na linguagem de especificação. Um mapeamento semântico de signos de interface tem por objetivo associar uma semântica elementar de maneira que permita ao usuário aplicar os widgets de uma ferramenta de design de interface de acordo com o seu significado preferencial.

Os widgets como tipos expressivos

Os widgets são os elementos de interfaces disponíveis nas ferramentas de design e programação de interfaces que permitem a criação de dispositivos virtuais de interação. Os widgets podem ser descritos abstratamente como tendo uma *aparência* e um *comportamento* (*look & feel*).

A aparência define as propriedades visuais essenciais de um widget que lhe dão a forma que o distingue de outros widgets. O comportamento determina os aspectos dinâmicos do widget que podem ser utilizados, por exemplo, para feedback ou animação. São a aparência e o comportamento que podem ser a expressão do signo de interface.

A aparência e o comportamento devem seguir um padrão especificado cujo objetivo é manter consistente as suas características em todos os sistemas que os utilizam. Um padrão consistente de widgets é aquele nos quais significados preferenciais são estabelecidos. Na maioria dos padrões e ferramentas de interfaces este significado é estabelecido informalmente através diretrizes que indicam como utilizar cada widget.

O widget se torna um signo quando se atribui um significado à sua aparência e ao seu comportamento. Através da sua aparência e comportamento, um único widget além de servir ao seu propósito de interação entre o usuário e o sistema, pode comunicar os diversos elementos do modelo de usabilidade, desempenhado a função metacomunicativa esperada de um *signo de interface*, como visto no capítulo anterior.

A estruturação semântica realizada no capítulo 5 e formalizada através da linguagem de especificação permitem a definição de tipos semânticos que podem ser correlacionados com os principais tipos de widgets, atribuindo-lhes um significado preferencial. De acordo com os conceitos semióticos vistos no capítulo 5, esta correlação funciona como um ***hipocódigo***, isto é, um **conjunto de regras de design incipientes que podem ser aplicadas na comunicação designer-usuario**.

Exemplos de widgets e suas equivalências

Existem inúmeros widgets propostos nos principais *padrões de interfaces*. Um padrão apresenta uma coleção de tipos de widgets cujo objetivo é manter consistentes a aparência e o comportamento (look & feel) de cada um deles entre as diversas aplicações.

A Tabela 0-3 apresenta os widgets de cinco conhecidos padrões de interface utilizados em diversas interfaces WIMP de sistemas interativos [Lee 93]. Esta tabela será utilizada como referência para a exemplificação de widgets quando apresentarmos as diversas categorias de nossa classificação. Quando ao longo do texto mencionarmos um widget que exista nesta tabela, seu nome em inglês e o número da linha na qual ele aparece, será colocado entre parênteses. Os nomes da tabela são mantidos em inglês para sabermos exatamente a qual widget de um determinado padrão estamos nos referindo.

	Apple	SAA/CUA	NextStep	Motif	OPEN LOOK
1	multiple windows, split window panes and panels	multiple-document interface, workplace, split windows	main window, standard windows	multiple primary windows, multiple window panes	multiple base windows, split panes, multiple panes
2	dialog boxes	dialog boxes	panels	dialog boxes	pop-up windows
3	menu bar, titles, pull-down menus, menu items	menu bar, menu bar choices, pull-down menu, menu items	main menu, submenus	menu bar, titles, pull-down menus, menu selections	control area, menu buttons, button menus, menu items
4	scrolling menus				
5	hierarchical menus	cascaded menus	submenus	cascading menus	submenus
6	pop-up menus	drop-down (scrolling) lists	pop-up list	options menus	abbreviated menu buttons
7		pop-up menus		pop-up menus	pop-up menus
8	tear-off menus		tear-off sub-menus	tear-off menus	menus with pushpins
9	palettes	value sets			exclusive settings
10	buttons	push buttons	push buttons	push buttons	buttons
11	check boxes	check boxes	switches (toggles)	check buttons	check boxes, non-exclusive settings
12	radio buttons	radio buttons	radio buttons	radio buttons	exclusive settings
13		Spin buttons		stepper butons	numeric fields
14	dials (progress indicators)	progress indicators	sliders	scales	sliders, gauges
15	Scroll bars	scroll bars	scrollers	scroll bars	scroll bars
16	1-D arrays	list boxes		list boxes	scrolling lists
17					hierarchical scrolling lists
18		combination boxes			
19	representation-type-in pop-up menus	drop-down combination boxes			text field with abbreviated menu buttons
20	2-D arrays		matrix		tables
21	Text fields	entry fields	text fields	text entry boxes	text fields
22			forms		
23		group boxes	group boxes		

24			pull-down lists		

Tabela 0-3: Equivalências entre nomes de widgets [Lee 93].

Os widgets podem ser utilizados para diversos propósitos comunicativos. Quando associamos os widgets aos tipos semânticos estamos definindo o repertório de tipos-signos de interface do sistema semiótico. O modelo de interação descrito no capítulo anterior e incorporado na linguagem de especificação oferece uma base para a classificação dos widgets mais comuns nos padrões e ferramentas de interfaces. Esta classificação de widgets é a base para a definição da semântica do sistema expressivo. Vamos apresentar a seguir nosso repertório de tipos-SI classificados de acordo com os tipos semânticos do modelo de usabilidade.

O nosso SSADIU apresenta os seguintes tipos-signos de interface:

- *Tipos-SI para mensagens de metacomunicação direta*
- *Tipos-SI de acionamento para ativação*
- *Tipos-SI de acionamento para fornecimento de informações para signos do domínio*
- *Tipos-SI para Visualização*
- *Tipos-SI para Mensagens de Comandos*
- *Tipos-SI para Mensagens de Tarefas*
- *Tipos-SI para Mensagens de Controle*

Vejamos cada um deles separadamente.

Tipos-SI para metacomunicação direta

Em nosso modelo denominamos de *mensagens de metacomunicação direta* aquelas que o designer envia para o usuário com o objetivo de realizar uma metacomunicação: comunicar algo sobre os elementos da interface destinados ao processo de interação usuário-sistema. Esta mensagem é direta por diferenciar-se das mensagens indiretas de metacomunicação que o designer envia através da forma e comportamento dos signos de interface.

A maneira mais simples que o designer possui para mandar um mensagem direta é através de **rótulos (labels)**. Os rótulos são signos de interfaces que podem ser widgets independentes, também chamados de **texto de leitura (read-only message)**, ou fazer

parte de outro widget, como por exemplo rótulos de **botões de comandos**, **janelas, caixas-check**, e diversos outros widgets.

Algumas mensagens diretas do designer podem ser expressas em janelas exclusivas visando chamar a atenção dos usuários sobre algum aspecto específico do processo de interação como, por exemplo, as **janelas de mensagem (message box ou notice)** que podem ser utilizadas para advertir ou informar o usuário.

Podemos ainda classificar as mensagens diretas do designer como possuindo três funções: *informação, realce e decorativos*. As de informação por sua vez podem ser utilizadas pelo designer para desempenhar os *atos de fala* do designer. Não discutiremos estas possibilidades pois atos de fala na comunicação designer-usuário geram assuntos complexos o bastante e necessitam ser discutidos num trabalho específico. Diremos apenas que dentre os informativos o designer pode fazer *asserções* (quando ele informa ao usuário sobre algo), *questões* (quando ele questiona ao usuário uma informação que pode ser utilizada para uma tomada de decisão pré-programada) e *comandos* (quando ele determina qual o ação o usuário deve fazer).

A função de *realce* é utilizada pelo designer para chamar a atenção do usuário sobre um aspecto específico da comunicação usuário-sistema. Esta função pode ser desempenhada por propriedades específicas que um signo de interface possa adquirir como uma cor mais forte, piscando, etc.

A função *decorativa* pode ser utilizada por critérios meramente estético por designer gráficos, mas também com propósitos comunicativos uma vez que aspectos estéticos abrem espaço para o desenvolvimento de *hipercódigos*, como discutimos no capítulo 5.

Tipos-SI de acionamento para ativação

Os signos para ativação devem permitir que ações do usuário ativem elementos do modelo de usabilidade com a execução e funções aplicadas da funcionalidade ou a apresentação de outras mensagens da interface.

São exemplos de signos de interface para ativação, o **botão de acionamento (10-Push-buttons)** - que pode ser utilizado para ativação de funções aplicadas - e de controles de janelas - que controlam a exibição de mensagens. (Figura 0-5).

Figura 0-5: Botões de Acionamento

A subdivisão entre acionamento de funções aplicadas e controle é importante por revelar aspectos semânticos distintos. Entretanto, na maioria das ferramentas ou padrões de interfaces não existem widgets distintos para elas. Assim estes critérios ficam apenas mencionados sem, no entanto, determinar novas categorias de widgets.

Tipos-SI de acionamento para fornecimento de informações

Os signos de acionamento para fornecimento de informações devem permitir ao usuário atribui-las a um signo do domínio. Estes signos podem ser classificados de acordo com o tipo de representação do signo do domínio. Apresentamos no capítulo 5 os principais tipos de dados previstos em nosso modelo. São eles:

Tipos Simples:

valores -verdade
numéricos
textos

Tipos Estruturados:

conjuntos
tabelas
hierarquias
redes

A seguir veremos exemplos dos principais tipos de signos de interfaces (tipos-SI) associados com os tipos de representações dos signos do domínio. A maioria deles possuem as funções de acionamento, para o fornecimento de informações, e de revelação que permite ver qual dado está atribuído ao signo do domínio.

A representações do tipo valor verdade podem ser fornecidos e visualizados por **caixas-check (11-checkboxes)**. Caixas-check representam condições binárias associadas a funções aplicadas que podem ser interpretadas como *sim/não*, *ligado/desligado*, *ativado/desativado*, etc.

O tipo de representação numérico corresponde ao conjunto de números inteiros ou reais. Pode-se utilizar um **campos numéricos (13-numeric fields)** ou **campos numéricos com incrementadores (13-spin buttons)**. Os incrementadores são elementos do widgets que permitem incrementar o valor numérico corrente do widget. Os tipos numéricos possuem variantes para as quais pode-se utilizar signos de interface mais específicos.

Um intervalo de inteiros corresponde a um conjunto de inteiros com um limite inferior e superior conhecidos. Para um *intervalo curto* de valores inteiros limitados pode-se utilizar como SI um **escorregador (14-Slider)** com labels numéricos correspondentes ao intervalo.

Intervalos longos de inteiros não devem ser acionados através de **escorregadores**, pois pode perder precisão de especificação. Neste caso deve-se utilizar **campos numéricos.**

Para os números reais, pode-se utilizar **campos numéricos** que aceitem reais ou mesmo **campos de textos,** quando campos numéricos não estiverem disponíveis, devido ao ponto decimal. Quando o tipo real pertencer a um conjunto de reais limitados em um intervalo curto e com precisão de poucas casas decimais, pode-se utilizar um **escorregador (14-slider)**.

Textos são seqüências de caracteres alfanuméricos e podem ser acionados em **campos de textos (21-textbox** ou **texfield).** Os campos de texto podem ser com uma **única-linha**, **múltiplas-linhas** e **múltiplas-linhas-com-barras**, que devem ser utilizados de acordo com o tamanho esperado do texto e da disponibilidade de espaço no medium interface.

Quando o tipo texto é de um conjunto pré-definido de informações do domínio ele possui a particularidade de que seus possíveis valores são específicos e possivelmente desconhecidos do usuário. É interessante, então, comunicar estes possíveis valores ao usuário de maneira que ele escolha dentre estes o que ele deseja atribuir ao signo do domínio. Existem diversos tipos-SI para esta categoria que dependem de três critérios:

número de elementos do conjunto = poucos ou muitos
variação no número de elementos = {estáticos ou dinâmico}

número de escolhas = única-exclusiva ou múltiplas
tamanho do maior elemento = {texto longo ou texto curto}

Quando se pode escolher apenas uma única opção e o número de elementos é pouco e não varia (estático) pode-se utilizar os **botões de opção (12-Radio-button** ou **option-buttons)**

Quando o número de elementos do conjunto for variável pode-se utilizar os diferentes tipos de **listas de seleção (16-list boxes)** que podem ser **com barra de rolamentos** se o número de elementos forem muitos; e com possibilidade de única ou múltiplas escolhas quando for o caso. Quando se pode fazer múltiplas escolhas e se tem um número de elementos *estáticos* (*layout*) e *poucos*, pode-se utilizar grupos de **caixas-check (11-checkboxes)**. As **listas pula-abaixo (24-Pull-down list)** e suas variações podem ser utilizadas nos mesmos casos das **listas de seleção** e devem ser aplicadas por questões estéticas ou por falta de espaço no medium interface. As **combinações campo-de-texto e lista (19-Combination ou 18-Drop-down)** são utilizadas quando o conjunto, independente do seu tamanho pode ser incrementado por mais um valor que o usuário digita. Uma outra aplicação é utilizar o campo de texto quando o usuário já sabe qual elemento ele deseja escolher.

Os tipos de representações estruturados requerem signos de interface mais sofisticados. Existem alguns pacotes de widgets comercializados que oferecem implementações de widgets para manipular listas ordenadas, tabelas, árvores (no Visual Basic existem *árvores de diretórios*).

As estruturas agregadas podem ser manipuladas por composições de widgets para cada tipo componente da estrutura agrupados em um **formulário** (que serão vistos adiante).

Tipos-SI para Visualização

Os signos de revelação têm por objetivo mostrar para os usuários os estados de funções aplicadas e os signos do domínio. Para os valores alfanuméricos os widgets **rótulo** (**label**) podem ser utilizados. Para os dados estruturados algumas ferramentas de interface proporcionam gráficos (**charts**) que podem ser tipo **torta, barra** ou **linha**, e para os dados bidimensionais as **tabelas (20-tables)**.

Signos de interface para revelar estados de funções aplicadas não são muito freqüentes na maioria dos padrões. O mais comum é o **indicador de progressão** que indica o percentual da tarefa realizada. Algumas interfaces mais sofisticadas implementam **ícones animados** como forma de revelar o estado da função aplicada. O ícone com uma impressora emitindo papel no MS Word é um exemplo de indicador do estado da função aplicada de impressão. O MS Windows apresenta a transferência de arquivos em uma janela com ícones representando papéis voando entre pastas associados a um indicador de progressão.

Configurações especiais e temporais de Tipos-SI

As categorias de tipos-SI apresentadas anteriormente aplicam-se às mensagens relacionadas com interações básicas. Para estas mensagens procuramos correlacionar um certo tipo de widget a uma determinada interação que o usuário necessita realizar. Vimos que as interações básicas podem ser articuladas. Para cumprir o propósito metacomunicativo argumentado pela Engenharia Semiótica o designer precisa comunicar ao usuário não apenas as interações básicas individuais, mas também comunicar as estruturas que permitem a ele desempenhar comandos e tarefas.

No *modelo de interface como expressão* apresentado no capítulo 5 vimos que signos de interface devem ser posicionados no medium de acordo com configurações espaciais e temporais. São as configurações de SIs que podem realizar a comunicação das estruturas de tarefas e ações. Existem alguns widgets que podem desempenhar o papel de signos de estruturação das mensagens permitindo ao designer realizar a *configuração espacial e temporal* dos signos no medium interface.

As *configurações espaciais* de widgets podem ser feitas através de alguns tipos de widgets disponíveis na maioria dos padrões de interface. **Janelas (1-Windows), caixas de diálogo (2-Dialog boxes), paineis de controles (2-control panel), formulários (22-Forms), quadros (23-frames) e menus (3-menus e 4-Pop-up menus)** são exemplos de widgets para estruturação espacial. As configurações espaciais, por permitir ao designer estruturar os widgets no medium interface, podem ser utilizadas de uma maneira geral para comunicar as estruturas de articulação *seleção, agrupamento* e *combinação.*

A maioria dos padrões de interface não apresenta widgets com *configurações temporais*. Normalmente elas são implementadas pelo programador da interface, ou por recursos de temporização que uma ferramenta venha a oferecer (Visual Basic). Vamos discutir rapidamente como as configurações temporais podem ser utilizadas na metacomunicação e sua aplicação na comunicação de mensagens estruturadas.

As *configurações temporais* podem ser utilizadas para comunicar as estruturas *seqüência* e *repetição* de ações ou comandos. Esta metacomunicação funciona de maneira a orientar ou direcionar as ações do usuário para obedecer uma determinada seqüência ou repetição.

Como as configurações temporais dizem respeito ao instante no qual um determinado signo de interface será apresentado ao usuário elas podem ser *baseadas-no-tempo* ou *baseadas-na-interação* com o usuário. No primeiro caso, o designer determina um processo de animação de uma seqüência de widgets. As configurações baseadas no tempo não são muito utilizadas para metacomunicação das estruturas de ações e tarefas. Seu principal uso é para estruturar mensagens de revelação *animadas* que podem ser utilizadas para a comunicação de estados de funções aplicadas e de mensagens diretas que podem ser utilizadas em tutoriais.

As *configurações temporais interativas* é que dão o caráter dinâmico e interativo da mensagem do designer. Elas devem ser especificadas pelo designer através da combinação de signos de interface para configuração espacial com os signos para mensagens de acionamento de controle.

As configurações temporais interativas podem ser especificadas por Diagramas de Estados Finitos, Redes de Petri ou outros formalismos baseados em eventos, nos quais as ações de usuário podem ser interpretadas como eventos que determinam novas ações do sistema possibilitando a apresentação dos widgets necessários ao desempenho de uma *seqüência* ou *repetição.*

Tipos-SI para Mensagens de Comandos

A estruturas discutidas acima são aplicadas tanto aos diferentes níveis de interação como às estruturas de controles de leitura da mensagem. As mensagens sobre as estruturas de articulação das interações básicas que compõem um comando podem ser

expressas pelos widgets **caixas de diálogo (2-Dialog boxes)**, **formulários (22-Forms) e quadros (frames)**.

Uma **caixa de diálogo** pode ser utilizada para uma *mensagem de comandos de função* que articule diversas interações básicas. Os **painéis de controle** devem ser utilizados para articular *seleção, combinação* e *agrupamento* de *interações básicas de acionamento.* Uma configuração temporal de painéis de controle permite *seqüências* e *repetição* de *interações básicas.* **Formulários** articulam *agrupamento* de *interações básicas* de fornecimento de *informações.*

Os **quadros** permitem *agrupamento* e *combinação* de interações básicas. O **quadro (frame)** pode ser utilizado como recurso auxiliar para agrupar signos de interface que pertençam a uma mesma categoria do domínio, enfatizando a possibilidade de seleção.

Recursos de layout também podem ser utilizados para comunicar seleções. Agrupamento e simetria são recursos da psicologia *Gestalt* bastante utilizados no design gráfico para comunicar relacionamentos entre elementos. Por exemplo, um agrupamento visual de botões de comando em uma caixa de diálogo podem comunicar uma *seleção de acionamento* dentre os possíveis acionamentos a serem desempenhados pelo usuário.

Normalmente as estruturas de interações que formam um comando devem ser rigorosas, isto é, o usuário precisa desempenhar as ações de acordo com a estrutura que foi determinada pelo designer. Neste caso os widgets utilizados (caixas de diálogo ou formulários) devem ser **modais**. Várias **caixas de diálogo modais**, isto é que requer interação exclusiva, podem ser utilizadas para articular uma *seqüência* de *mensagens de comandos* que façam parte de uma *mensagem de tarefa.* As estruturas de ações *seqüência* e *repetição* podem ser comunicadas por *configurações temporais interativas* de **caixas de diálogo modais**. Uma sucessão de caixas de diálogo contendo um widgets cada uma podem indicar a estrutura *seqüência* de ações que o usuário necessita desempenhar para acionar um comando. A *repetição* é semelhante entretanto deve-se apresentar a mesma caixa de diálogo.

Tipos-SI para Mensagens de Tarefas

As **janelas (1-Windows)** podem ser utilizadas para as *mensagens de tarefas* estruturadas com *agrupamento* e *combinação* de *comandos, controles de leitura* e *visualizações.* Diversos grupos de tarefas podem ser colocados em diferentes janelas. Todas as janelas podem ser agrupadas em uma janela principal (janelas MDI, no MS Windows).

O agrupamento de tarefas em janelas podem ser utilizados para comunicar relações entre os funções aplicadas e entre as funções aplicadas de um sistema. As funções aplicadas que atuam sobre um mesmo signo do domínio podem ser agrupadas em uma mesma janela.

Tipos-SI de Mensagens de controles

Muitas vezes as **janelas** (que apresentam mensagens de tarefas), **caixas de diálogo** ou **formulários** precisam ser acionados por *controles de mensagens.* Estes controles de mensagens devem estar presentes nas mensagens de tarefas e de comandos. Muitas vezes o sistema possui dezenas de comandos num mesmo grupo de tarefas e não podem ser apresentados todos de uma vez numa só mensagem de tarefa. Eles precisam ser apresentados ao usuário de maneira organizada. Os acionadores de controles de mensagens devem ser comunicados em **menus**.

Os **menus** são signos de interfaces destinados à *seleção* de *interações básicas* de *acionamento de controles e de operações.* Eles podem ser utilizados na apresentação dos comandos ou controles de comandos permitindo que o usuário selecione o que deseja realizar.

Os menus podem ser **lineares** (**listas simples ou 4, 6 e 7 pop-up menu**) ou **hieráquicos (3 e 5 Menu bar/Pull-down)** e devem ser escolhidos pelo designer de acordo com o número de comandos ou de controles de mensagens de comandos que ele necessite apresentar ao usuário. Menus lineares articulam interações para seleção. **Menus pop-up** são utilizados para o aninhamento de *seqüência* com *seleção.*

Regras de Correlação (o mapeamento semântico)

Na seção anterior procuramos mostrar como os diversos widgets dos principais padrões de interfaces existentes podem mapeados no principais elementos do nosso

modelo de usabilidade. Com esta correlação é possível escolher os widgets de acordo com a função que ele pode desempenhar no processo de meta-comunicação.

A Figura 0-6 esquematiza o papel das regras de correlação no mapeamento semântico entre os termos da LEMD e as ferramentas de interface.

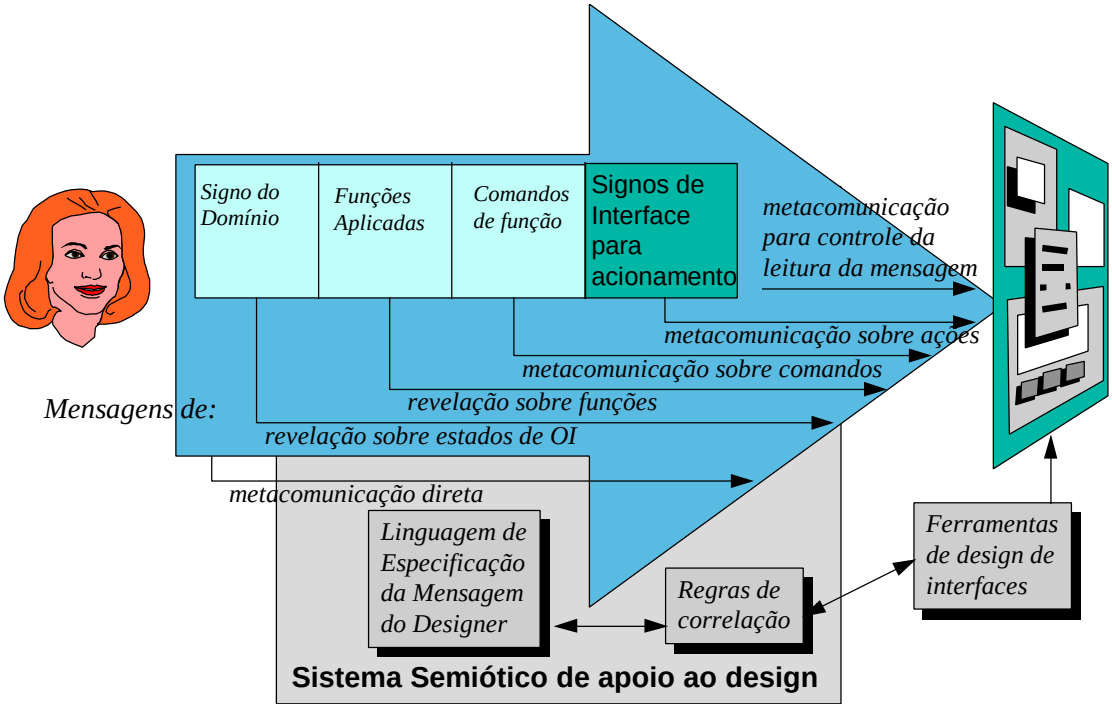


Figura 0-6 : O papel das regras de correlação no SSADIU.

Nesta seção apresentaremos um resumo das regras de correlação entre os signos de interface - os tipos expressivos do sistema semiótico - e os elementos do modelo de usabilidade descritos através da linguagem de especificação - os tipos semânticos do sistema semiótico.

As regras de correlação estão descritas na Tabela 0-4. Elas visam descrever os significados esperados para os principais signos de interface. Como o modelo de usabilidade é o objeto da mensagem de interface, o processo de estruturação realizado no capítulo anterior e descrito através da linguagem de especificação da mensagem de design (*segmentação do contínuo do objeto*) descreve os *tipos semânticos* aos quais os tipos expressivos precisam ser associados. Esta correlação é feita por um mapeamento entre os principais widgets e os termos da linguagem de especificação. Como um tipo semântico é caracterizado por sentenças da linguagem de especificação, o seu

vocabulário determina apenas *tipos semânticos parciais*. O signo de interface a ser associado a um termo da linguagem depende do restante da sentença na qual ele aparece. Estas considerações estão descritas na tabela. Os termos da linguagem para a especificação de signos do domíno ou de funções aplicadas não constituem mensagens e não possuem tipo expressivo associado.

Termos da Linguagem (tipos semânticos parciais)	**Widgets (tipos expressivos)**
`Activate`	Um **Botão** é o tipo-SI básico para mensagens de acionamento.
`Application-Function`	**Rótulo texto ou ícone** associado com os widgets de mensagem de comando ou de estado de função.
`Available`	**Rótulo texto ou ícone** que indique estado disponível de função. Pode-se utilizar a cor do rótulo para indicar a disponibilidade.
`Combine`	**Quadro** ou *layout de agrupamento* de widgets
`Command`	**Apresentação** do **SI** correspondente ao comando.
`Command-Message`	**Caixa de diálogo**, **painel de controle** ou **formulário**.
`Continue`	**Rótulo texto ou ícone** associado a botão de acionamento que indique continuar.
`Control`	Não é mensagem.
`Control-Message`	**Menus**
`Discard`	**Rótulo texto ou ícone** associado com botões de acionamento que indique fechar
`Domain-Sign`	O signo do domínio possui um nome que associa um conceito do domínio com uma informação específica. O conceito é expresso por um **rótulo texto**. A informação específica depende do tipo de representação.
`Enter`	**Campos de texto, Campos numéricos, Campos numéricos com incrementadores.**
`Finished`	**Rótulo texto ou ícone** que indique estado finalizado de função.

for Application-Function	**Rótulo texto ou ícone** associado a botão com nome da função num widget de mensagem de comando.
Hierarchy-of	**Widgets de visualização de Árvores**
Information-of	**Listas, Tabelas, Árvores, Grafos, Gráficos, Campos de texto (somente-leitura), Campos numéricos (somente-leitura)**.
Join	**Configuração espacial** de widgets numa caixa de diálogo, formulário ou painel de controle.
Maximize	**Rótulo texto ou ícone** associado a botão de acionamento que indique maximização
Minimize	**Rótulo texto ou ícone** associado a botão de acionamento que indique minimização
Network-of	**Widgets de visualização de Grafos**.
Number	**Caracteres numéricos**
Operands	Não é mensagem.
Pre-condition	Não é mensagem.
Post-condition	Não é mensagem.
Repeat	**Configuração temporal** de um mesmo widget.
Representation-type	Não é mensagem.
Running	**Ícone animado**
Select	**Menus** ou **Configuração espacial** de widgets numa caixa de diálogo, formulário ou painel de controle, para seleção de acionamentos. **Lista de seleção, Listas pula-abaixo, Caixas de combinação, Grupos de caixas-check** para seleção de informações
Sequence	**Configuração temporal** de widgets. **Caixas de diálogo modais**.
Set-of	**Lista de seleção, Listas pula-abaixo, Caixas de combinação, Grupos de caixas-check.**
Show	**Rótulo texto ou ícone** associado a botão que indique ativação de mensagem.
Start	**Rótulo texto ou ícone** associado a botão de acionamento

	que indique iniciar função.
`State-of`	**Rótulo texto ou ícone** que indique estado de função.
`Step-through`	**Rótulo texto ou ícone** associado a botão de acionamento que indique executar passo-a-passo.
`Stop`	**Rótulo texto ou ícone** associado a botão de acionamento que indique finalizar a execução.
`Suspend`	**Rótulo texto ou ícone** associado a botão de acionamento que indique suspender a execução.
`Suspended`	**Rótulo texto ou ícone** que indique estado suspenso de função.
`Table-of`	**Widget Tabela**.
`Task-Message`	**Janela Simples ou MDI.**
`Text`	**Caracteres alfanuméricos.**
`Truth-value`	**Check boxes.**
`Unavailable`	**Rótulo texto ou ícone** que indique estado suspenso de função.
`Undo`	**Rótulo texto ou ícone** associado a botão de acionamento que indique desfazer função.
`View`	**Listas, Tabelas, Árvores, Grafos, Gráficos, Campos de texto (somente-leitura), Campos numéricos (somente-leitura)** para informações, e **Rótulo texto ou ícone** que indique estado de função.
`Waiting-for-input`	**Rótulo texto ou ícone** que indique estado esperando de função.
`Waive`	**Rótulo texto ou ícone** associado a botão de acionamento que indique cancelar função.

Tabela 0-4: Regras de correlação semântica entre os tipos semânticos determinados pela linguagem de especificação e os tipos expressivos das principais ferramentas de interface.

Resumo

O SSADIU apóia o design oferecendo um repertório de tipos-signos que o designer pode instanciar na elaboração do objeto e da expressão da mensagem. Os tipos semânticos estão descritos na forma de uma linguagem de especificação do modelo de

usabilidade. Os tokens semânticos são portanto sentenças construídas de acordo com as regras desta linguagem. Os tipos expressivos são os widgets disponíveis na maioria das interfaces. A figura 0-7 mostra os componentes do SSADIU.

A LEMD permite uma melhor formulação do modelo de usabilidade a ser comunicado. Ela fornece ao designer recursos para a estruturação da mensagem que revelam as distinções conceituais do modelo de usabilidade. As estruturas possibilitam ainda que tenhamos um mapeamento entre elas e os tipos expressivos.

As regras de correlação semântica permitem que os tipos de widgets utilizados de acordo com o seu significado preferencial.

As sentenças da LEMD, por serem imperativas, podem indicar que o designer esteja impondo o que o usuário deve fazer não permitindo que ele possua o controle da interação. No entanto, o que a nossa abordagem revela é que é sempre o designer quem toma as decisões de tudo o que é comunicado na interface inclusive o aquilo que o usuário pode fazer (o modelo de interação). As *estruturas de seleção* da linguagem e os mecanismo de *controle de mensagem* oferecem toda a flexibilidade necessária para o usuário controlar o processo de interação e de leitura da mensagem do designer.

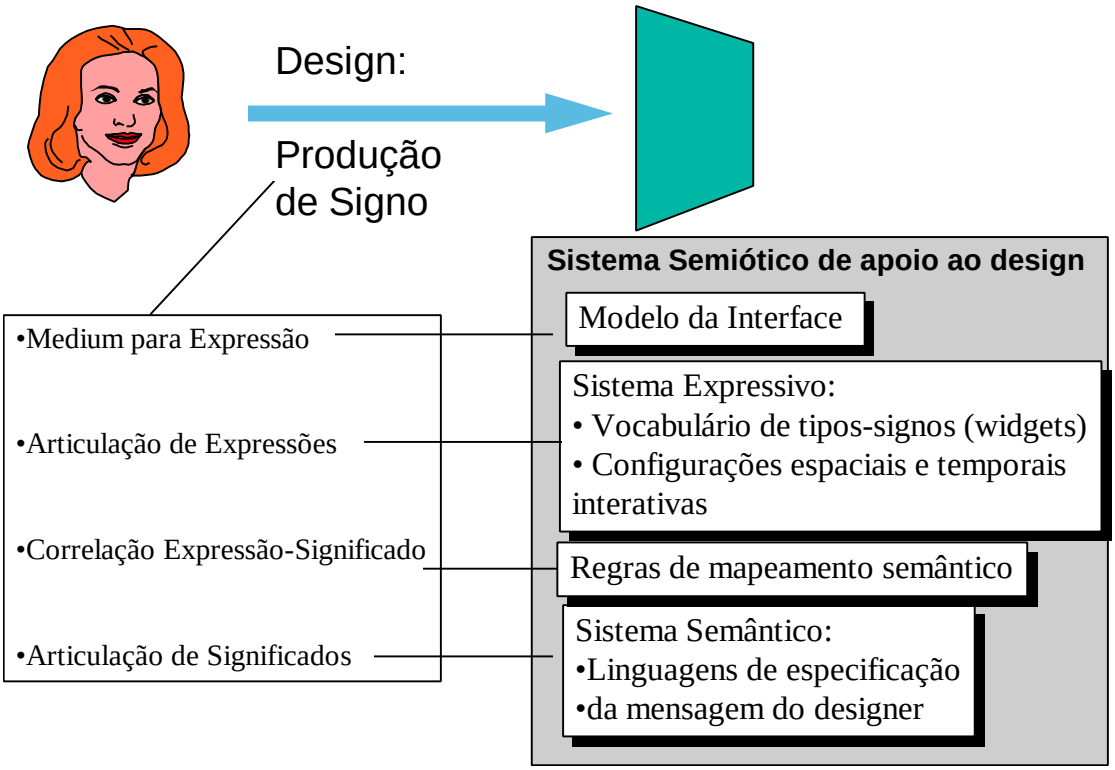


Figura 0-7: Componentes do SSADIU

CONCLUSÃO

Com a motivação do *desafio de usabilidade* proposto por [Adler & Winograd 92] e pelas perspectivas que consideram sistemas computacionais interativos como ferramentas intelectuais e *medium*, a Engenharia Semiótica apresenta a perspectiva de que eles são artefatos de metacomunicação. Nesta abordagem eles devem ser projetados como uma *solução-em-potencial* a ser comunicada através da sua interface. A solução-em-potencial é o *modelo de usabilidade* que determina o que o usuário pode fazer (funcionalidade) e como ele pode utilizar o sistema (modelo de interação) a partir da interpretação que o designer tem dos seus problemas no domínio de aplicação do sistema. O designer, portanto, envia uma mensagem para os usuários através do sistema cuja expressão é a interface e o conteúdo é o modelo de usabilidade.

Nossa abordagem partiu desta perspectiva com o objetivo de apresentar explicações, previsões e instrumentação através de modelos para o processo de interação entre usuários e sistemas e ferramentas de apoio ao design de interfaces como atividades integrantes de um processo metacomunicativo mais geral. Limitando nosso escopo de atuação ao design de interfaces GUI/WIMP nosso trabalho partiu da perspectiva de metacomunicação da Engenharia Semiótica [de Souza 93] e, baseado nos conceitos de *signo* e *semiose ilimitada* de Peirce [Peirce 31] e na teoria semiótica de Eco [Eco 76], apresentou como contribuições:

um **modelo para o processo de design como produção de signos** a partir de conceitos da teoria semiótica de Eco [Eco 76], integrado ao processo de design e desenvolvimento tradicional do sistema;

um **meta-modelo para o Modelo de Usabilidade como objeto da Mensagem do Designer** que descreve quais componentes funcionais e de interação o usuário precisa aprender;

um **modelo da interface como expressão da Mensagem do Designer** que descreve os seus componentes e como eles podem ser utilizados para desempenhar as funções de acionamento, revelação e metacomunicação.

um **Sistema Semiótico de Apoio ao Design de Interfaces de Usuário** que oferece um formalismo (**uma Linguagem de Especificação da Mensagem do Designer**) para a elaboração do modelo de usabilidade

a ser expresso através da interface cujas sentenças podem ser mapeadas semanticamente através de **regras de correlação** a widgets das principais ferramentas de interfaces.

Comentários Sobre as Contribuições

Os beneficiários diretos de nosso trabalho são pesquisadores de IHC, designers e usuários. Os pesquisadores dispõem de modelos teóricos com explicações sobre fenômenos da interação usuário-sistema que apresentam perspectivas alternativas às das abordagens cognitivas, que são maioria em IHC. O conceito de signo, de interpretante e de semiose ilimitada revela o importante papel que os sistemas semióticos desempenham na aquisição de conhecimento e a limitação que os modelos cognitivos possuem por não abordar tal processo como um processo semiótico. Os pesquisadores dispõem ainda de modelos dos componentes, de funcionalidade e de interatividade que podem ser utilizados na construção de outras ferramentas.

Os designers também são beneficiários diretos por utilizarem um formalismo complementar àqueles que normalmente dispõem no design da interface. A LEMD permite-lhes formular sua mensagem de maneira estruturada e coerente com o meta-modelo de usabilidade. Este meta-modelo descreve os componentes necessários ao desempenho do processo de interação. O formalismo é complementar às linguagens de especificação tradicionais e pode ser integrado às ferramentas de inteface baseadas em widgets, normalmente utilizadas para intefaces GUI/WIMP.

O usuário, além de ser um beneficiário indireto quando utiliza sistemas cujas interfaces produzidas com o apoio do SSADIU são mais fáceis de intepretar, é também um beneficiário direto quando utiliza sistemas ciente da perspectiva de metacomunicação. Neste caso, ele utiliza o sistema conhecendo as diferenças conceituais do modelo de usabilidade e espera consistência na utilização dos signos de interface.

O modelo para o processo de design como produção de signos

Na Engenharia Semiótica o processo de design é visto como uma atividade semiótica, isto é, que inclui processos de produção de signos (expressão, significado e interpretante). Entretanto, este processo não deve ser visto com uma atividade comunicativa simples. Ele é essencialmente uma metacomunicação indireta e unidirecional, uma vez que a mensagem que é comunicada se compõe de um sistema semiótico para a interação usuário-sistema que é apresentado ao usuário em tempo de

execução. O que se comunica é um modelo de funcionalidade e de interação para usuários explorarem o sistema.

A Engenharia Semiótica requer que este processo seja caracterizado como uma atividade de *produção de signos e sistemas de comunicação* dentro de um processo comunicativo. A teoria semiótica de Eco oferece os fundamentos necessários para esta abordagem através de conceitos da *teoria dos códigos e da teoria da produção de signos*. Neste processo existem as atividades de formulação do conteúdo a ser comunicado e a sua representação em algum sistema expressivo. Estas atividades devem ser guiadas por um sistema semiótico de maneira que se tenha uma ferramenta que apóie o designer neste processo comunicativo.

A proposta de Engenharia Semiótica original [de Souza 93] aplicou a *teoria da produção de signos* na elaboração de *diretrizes* que auxiliam o designer na escolha dos signos da interface de acordo com 4 parâmetros de produção de signos. Nosso trabalho seguiu um outro caminho na tentativa de encontrar um formalismo para especificação que pudesse ser integrado a um processo de design. Combinando conceitos da teoria dos códigos e da produção de signos apresentamos um modelo teórico descreve a aplicação como um signo formado por signos. Cada um destes signos pode ser produzido utilizando diversos códigos e subcódigos que se superpõem e se articulam entre si num processo de *extra-codificação*. Este processo requer a utilização de *tipos-signos* que são estruturas pré-definidas de signos que já incorporam os diversos códigos necessários a sua produção e interpretação.

Baseados neste modelo desenvolvemos um sistema semiótico que sistematiza os diversos códigos utilizados no design da interface. No SSADIU, ao invés de diretrizes o que temos são regras que correlacionam o modelo de usabilidade de um sistema com os widgets. Estas regras de design são um conjunto de códigos incipientes (hipcódigos) e sobrepostos (hipercódigos) que permitem uma interpretação consistente

Os conceitos de semiótica determinaram como requisitos a necessidade de se fazer uma estruturação da interface e do modelo de usabilidade que são a expressão e o objeto da mensagem do designer. O modelo apresentado para a interface permite ao designer construir uma expressão através de signos de interface que são os elementos

com os quais o usuário pode interagir. Estes signos são dispostos espacial e temporalmente no *medium interface*, e podem ser manipulados por ferramentas de acionamento. Este modelo da interface oferece recursos suficientes para a elaboração da mensagem dinâmica, interativa e computável.

O modelo de usabilidade como objeto descreve a funcionalidade em termos de signos do domínio e de funções aplicadas que os modificam. As funções aplicadas devem ser ativadas por comandos construídos pelas ações realizadas pelo usuário usando os dispositivos de interação. Representações devem ser utilizadas para que o usuário visualize os elementos do sistema. Comandos, visualizações, funções aplicadas e signos do domínio são os elementos essenciais da usabilidade.

Este modelo distingue o que é essencial para a utilização daquilo que é metacomunicação que o designer envia para o usuário com o objetivo de melhorar a usabilidade do sistema. O designer pode realizar metacomunicação direta ou indireta a respeito dos elementos fundamentais e da própria interface. A comunicação a respeito da própria interface e a interação que o usuário realiza apenas com a interface faz parte daquilo que chamamos de *leitura interativa da mensagem do designer.*

O processo de interação, normalmente visto como um processo de diálogo entre o usuário e o sistema, deve ser interpretado e especificado por uma perspectiva diferente. Na Engenharia Semiótica o designer concebe o modelo de usabilidade como seus elementos fundamentais e deve realizar comunicações a respeito de comandos de funções aplicadas, tarefas, ações, estados do sistema. A maior parte dos elementos da interface, como por exemplo menus, rótulos, ícones que representam estados prévios do sistema, estão a serviço da metacomunicação. Uma pequena parte dos widgets é que são essenciais para a utilização do sistema.

Um exemplo disto é a navegação através de menus e janelas que o usuário precisa realizar para acionar um comando. Todos os acionamentos com o menu ou com os botões de controle de janelas são supérfluos para a interação. Apenas o widget correspondente ao comando é essencial. Contudo, todos eles exercem importância para a usabilidade.

O sistema semiótico de apoio ao design de interfaces de usuário

Nosso modelo mostra que os padrões e ferramentas de interfaces podem ser melhor utilizados se considerarmos os widgets como potenciais signos de interface (tipos-signos). Para isto é necessário tratá-los com parte de um sistema semiótico que os considere como tipos expressivos associados a tipos semânticos. Os tipos semânticos são instrumentos para a especificação do modelo de usabilidade.

O SSADIU permite a especificação do modelo de usabilidade como artefato de metacomunicação utilizando um repertório de tipos de signos de interface (widgets) estabelecidos por códigos que mapeiam suas propriedades expressivas (aparência e comportamento) em unidades semânticas. Os tipos-signos dão significados aos widgets dos principais padrões e ferramentas de interface, estabelecendo um mapeamento entre eles e as estruturas da linguagem de especificação do modelo de usabilidade.

A linguagem de especificação revela distinções conceituais importantes. Ela também orienta o designer a especificar o modelo de usabilidade como um processo de metacomunicação na qual ele deve comunicar os elementos da usabilidade através de mensagens abstratas.

A linguagem enfatiza ainda como os aspectos dinâmicos e de interatividade da mensagem do designer são essenciais para a sua interpretação pelo usuário, indicando que as abordagens voltadas para os aspectos estáticos e de layout da interface são limitadas.

Embora as ferramentas de design baseadas em widgets permitam que o designer se utilize dos mesmo elementos que o usuário no processo de interpretação da mensagem, elas não determinam como estes widgets podem ser articulados nem, principalmente, quais são os seus significados. O SSADIU complementa estas ferramentas mostrando como os widgets podem ser mapeados em sentenças da linguagem que especificam de maneira abstrata a mensagem que o designer quer enviar para o usuário. Esta linguagem apresenta estruturas que descrevem como as mensagens podem ser articuladas, permitindo a descrição de widgets compostos por outros widgets e quando se deve utilizar de configurações espaciais e temporais.

A perspectiva da Engenharia Semiótica para a usabilidade

Na perspectiva cognitiva, a usabilidade é a vista como sendo a qualidade da interface de proporcionar o menor esforço cognitivo para o usuário interagir com o sistema. Modelos cognitivos de competência e performance do usuário oferecem as métricas para medir este esforço e permitem avaliar a usabilidade da aplicação através da interface.

A Engenharia Semiótica faz esta avaliação analisando a mensagem do designer como signo. A interpretação do sistema através dos signos veiculados durante a interação permite ao usuário adquirir um interpretante. Pelo conceito de semiose ilimitada o interpretante é um processo, uma cadeia de significados ilimitados. Assim, o *modelo conceitual de usabilidade do usuário* não é constante, mas se altera a cada nova interação com o sistema num processo *ilimitado* e *não-determinístico* que possibilita ao usuário um agir criativo. Entretanto, ele está sujeito a convergência e estabilização ao longo do processo de interação/interpretação, uma vez que o sistema interativo é *determinístico.* A estabilidade é necessária para que o usuário adquira a competência necessária para utilizá-lo.

Assim, não é apenas a comunicação da funcionalidade e do sistema de interação que garantem a usabilidade. O sistema de interação deve ser projetado como uma mensagem que represente (seja um signo) uma solução em potencial para que o usuário desempenhe suas tarefas. A expressividade (o quanto pode ser representado) e grau de articulação do sistema de interação devem ser os parâmetros complementares que a Engenharia Semiótica oferece para análise e design de sistemas visando usabilidade.

Implicações da perspectiva de metacomunicação da mensagem do designer

Nosso trabalho apresentou um modelo para o sistema interativo como um signo, tal como conceitualizado por [Peirce 31], seguindo a perspectiva de metacomunicação apresentada em [de Souza 93]. Este modelo traz algumas contribuições fundamentais que serão agora discutidas.

No nosso modelo a interface tem o papel de ativar os interpretantes do usuário a respeito daquilo que o ele pode fazer - a funcionalidade, especificada em termos de

signos do domínio e de *funções aplicadas* - e de como ele pode fazê-lo - o modelo de interação, especificado em termos de *comandos, interações básicas e visualizações*. Cada elemento da interface tem o potencial de ser um signo. As palavras, ícones, widgets, ações do sistema e todos os elementos da interface podem adquirir significados para o usuário.

A mensagem do designer é dinâmica e interativa. O usuário não é um receptor passivo, mas um participante ativo da mensagem que está sendo enviada. Através dos atos de interação (as mensagens trocadas entre o usuário e o sistema) o usuário adquire o modelo de usabilidade. O usuário está desempenhando um duplo papel. Ele é agente emissor e receptor no processo de interação com o sistema e é receptor da mensagem que o designer envia para ele através da interface.

O usuário apenas se dá conta deste processo metacomunicativo quando ocorrem problemas durante a interação. Nestes casos, o usuário pode se perguntar: *"mas o que isto quer dizer?"* ou *"com eu posso fazer isto ou aquilo?"*. Nestas ocasiões ele sabe que tem que recorrer a manuais ou sistemas de ajuda onde está a palavra do designer dizendo a ele como resolver seus problemas.

A Engenharia Semiótica argumenta que é papel do designer colocar o usuário a par deste processo metacomunicativo indireto (que ocorre através dos elementos de interface) utilizando mensagens de metacomunicação diretas a serem enviadas também através da inteface. Designers normalmente realizam este processo metacomunicativo, sem se darem conta, quando buscam escolher uma melhor palavra para o rótulo de um widget ou quando querem escolher o widget mais adequado. Outras vezes, no entanto, por desconsiderar este processo, os designers causam problemas de interpretação e conseqüentemente diminuem a usabilidade do sistema.

O modelo da *aplicação como signo* estende a proposta original de [de Souza 93] e explica porque os métodos tradicionais da engenharia de software apresentam limitações para o design para usabilidade. Nestes métodos, o designer ou programador utiliza linguagens de especificação que são diferentes do tipo de linguagem que os usuários utilizarão para interpretar a mensagem do designer. Uma linguagem de programação ou de especificação geram expressões diferentes daquela que o usuário

interpretará e, consequentemente, o signo será diferente. Designer e usuário devem utilizar-se de um sistema semiótico comum ou, no mínimo, compatíveis, uma vez que eles fazem parte de um mesmo processo comunicativo.

A semiótica de Peirce enfatiza, através dos conceitos de interpretante e semiose ilimitada, que o processo de aquisição de conhecimento e o de comunicação em particular são não-determinísticos, uma vez que ele depende do conhecimento prévio e da situação na qual se encontram os participantes. Interagir requer adquirir o conhecimento necessário sobre o sistema interativo. Projetar esta máquina requer que se comunique sua usabilidade associando as intenções do designer com as dos usuários. Esta máquina virtual de software se caracteriza por uma natureza essencialmente simbólica, formada por "signos artificiais culturais" que devem estar inseridos no contexto específico daqueles usuários.

O modelo baseado no conceito de signo proporciona uma visão pragmática do processo de design deslocando o foco de interesse que inicialmente era centrado-no-sistema, nas abordagens da ciência da computação, e depois passou para centradas-no-usuario, nas abordagens cognitivas, para uma abordagem que considera o sistema, o usuário e o domínio de aplicação, todos juntos como parte de um processo semiótico. O sistema interativo como signo é um processo ilimitado que é instaurado através do processo de interação usuário-sistema. O design do sistema e de sua interface é parte de um processo evolutivo e cultural da sociedade. As inovações em interfaces não podem ser radicais, mas é essencial uma evolução que considere a cultura e o conhecimento prévio de seus usuários.

O modelo também explica uma importante limitação. Não é possível atingir o *determinismo* no design de interfaces. Devido ao processo de semiose ilimitada, não se pode garantir que os signos utilizados na interface sejam sempre interpretados pelos usuários da mesma forma pretendida pelos designers. O processo de design e de avaliação da usabilidade deve sempre considerar a possibilidade de múltiplas e ilimitadas interpretações. O modelo de usabilidade deve ser concebido e especificado como um modelo *aberto,* passível de extensões pelo usuário, através de utilização criativa e não previstas. Isto é possível porque a aquisição do modelo conceitual de usabilidade é

dinâmica e ilimitada, pois é um processo semiótico. O papel de um sistema semiótico de tipos de signos é tentar restringir este processo de interpretação em torno do modelo de usabilidade.

Comparação com Outras Abordagens

A Engenharia Semiótica é a denominação que caracteriza a atividade de design de sistemas interativos considerando-os com artefatos de metacomunicação através dos quais designer enviam um mensagem ao seus usuários. Ela não é uma engenharia no sentido de um método teoricamente fundamentado para a construção de todos os seus elementos componentes, mas de atividades necessárias à produção de sistemas de signos. Ela é uma abordagem que aplica conceitos de semiótica para a produção de signos no desenvolvimento de aplicações de software interativas.

Embora a aplicação de semiótica em computação, em IHC e em design de interfaces tenha sido abordada e justificada por diversos autores, conforme apresentado no capítulo 3, nenhum deles contrasta a semiótica com as abordagens tradicionais para IHC. A Engenharia Semiótica deve ser considerada como uma abordagem teórica complementar para IHC. É importante contrastar as diferentes abordagens entre si de maneira a comparar e ressaltar as diferenças e vantagens da Engenharia Semiótica perante as outras, especialmente a cognitiva que é a mais tradicional.

A Semiótica Computacional de Andersen apresenta uma análise e uma estruturação bastante amplas do sistema computacional, da interface e do contexto de utilização [Andersen 90]. Embora bastante útil para análise de interface, ela não apresenta técnicas, métodos e ferramentas que possam ser aplicadas diretamente no design. Sua classificação de *signos baseado em computadores* é reveladora para os aspectos de interação, mas muito pouco contribuem para a sua aplicação em design, uma vez que não são apresentadas regras que indicam como os signos devem ser utilizados. Nossa abordagem diferencia-se da Semiótica Computacional principalmente por enfatizar o potencial comunicativo que cada signo de interface tem e por mostrar como ele pode ser utilizado na prática de design.

Segundo a perspectiva cognitiva, a interação usuário-sistema corresponde às atividade físicas e mentais que possibilitam ao usuário desempenhar tarefas no sistema. O

foco nesta perspectiva são as atividades mentais que são vistas como um mapeamento entre dois modelos conceituais: o modelo do domínio da aplicação e das tarefas com o modelo conceitual que o usuário tem do sistema (interface e funcionalidade). Este mapeamento ficou conhecido como *Mapeamento Tarefa-Ação*. No modelo de Norman o mapeamento é analisado através dos golfos da execução e da avaliação e pelas distâncias articulatórias e semânticas. Norman argumenta que a *imagem do sistema*, sua interface de usuário principalmente, é a responsável por fazer o usuário adquirir os modelos mentais sobre o sistema. Comparando com o modelo de Norman, o *modelo do sistema* é o nosso *modelo de usabilidade*, os modelos conceituais do designer e do usuário são os modelos conceituais de usabilidade do designer e do usuário e *a imagem do sistema* é a nossa proposta de *mensagem do designer*.

As soluções apresentadas para o problema de design nas abordagens cognitivas consideram que conhecendo o processo de mapeamento tarefa-ação através de modelos cognitivos genéricos é possível determinar as características que devem ter a interface para garantir uma melhor usabilidade. Estes modelos cognitivos permitem ainda realizar previsões sobre o desempenho humano na interação com uma determinada interface.

A Engenharia Semiótica não aborda diretamente o mapeamento tarefa-ação e considera válida a perspectiva do problema da interação usuário-sistema como sendo um mapeamento entre estes dois modelos conceituais. Além disso, a Engenharia Semiótica não apenas enfatiza o papel da interface de usuário em fazer o usuário adquirir o modelo conceitual sobre o sistema, como apresenta uma perspectiva de comunicação para este processo. A idéia lançada por Norman é vista na Engenharia Semiótica como um processo de metacomunicação fundamentado por teoria semiótica.

Para que o usuário realize o mapeamento tarefa-ação a perspectiva cognitiva considera que é preciso aprender sobre a funcionalidade e o modelo de interação - isto é adquirir o modelo de usabilidade. As abordagens cognitivas buscam determinar modelos para este aprendizado bem como para os processo da execução e avaliação (ver o modelo de Norman [Norman 86]). Este processo de aprendizado é, segundo a perspectiva da Engenharia Semiótica, um processo de interpretação do modelo de usabilidade através do mensagem do designer.

Desta forma, o objetivo principal do designer deve ser o de construir uma mensagem que permita uma interpretação mais produtiva do modelo de usabilidade. Obviamente, estas duas abordagens são complementares, pois uma descrição precisa de como ocorre o processo de aprendizado poderia facilitar a construção da mensagem do designer. Entretanto, como não se sabe ainda como é este processo, a alternativa da Engenharia Semiótica que se utiliza de códigos conhecidos pelo usuário é mais viável, uma vez que os códigos visam orientar os processos cognitivos do usuário.

Segundo Norman, o modelo mental do usuário é informal, incompleto e inconsistente enquanto que o modelo do sistema é formal, completo e consistente. Na Engenharia Semiótica esta diferença fica explícita através das noções de interpretante e objeto do signo. O interpretante (modelo conceitual de usabilidade do usuário) é uma aproximação do modelo de usabilidade que é formal, completo e consistente, uma vez que é definido formalmente pela semântica de uma linguagem de programação.

A abordagem da Engenharia Semiótica é primeiramente uma mudança de paradigma e de prioridades que precisa ser complementada por abordagens cognitivas e métodos práticos de design. Enquanto que a abordagem cognitiva proporciona um perspectiva individualista cujo foco está no processo de aprendizado, compreensão e outros processos mentais, não importando quais as qualidades intrínsecas do que se deseja compreender (o usuário), a abordagem semiótica proporciona uma perspectiva interpessoal com foco no processo de interpretação e expressão de significados através de processos comunicativos, não importando as características subjetivas e individuais das pessoas envolvidas. As abordagens semióticas complementam as cognitivas considerando que o processo de interação usuário-sistema é parte de um processo mais global de comunicação entre o designer e o usuário.

A principal diferença da abordagem cognitiva para a Engenharia Semiótica é que o processo de design da primeira é guiado por modelos cognitivos genéricos enquanto que na segunda o design é guiado por sistemas semióticos de design estabelecidos, pragmaticamente, no contexto de uso e na cultura do usuário.

Vamos agora situar as principais contribuições do nosso trabalho dentro da tabela apresentada no capítulo 3 que sistematiza os diversos critérios para avaliação das abordagens cognitivas para IHC.

Categoria da abordagem	Sub-categoria da abordagem	Fonte teórica	Modelos teóricos e perspectivas	Contribuições para o design	Principais vantagens e limitações
Abordagem semiótica	Engenharia semiótica	Semiótica de Peirce e Eco	O sistema computacional é artefato de metacomunicação que conduz a mensagem do designer. O seu design é um processo de produção de signo	Um sistema semiótico que contempla uma linguagem de especificação do modelo de usabilidade numa perspectiva de metacomunicação e regras de correlação com widgets de interfaces.	Oferece apoio ao processo de construção da interface como mensagem, mas não oferece métodos para o design do modelo de interação e de funcionalidade.

A Avaliação dos Formalismos

A nossa abordagem gerou produtos a serem aplicados tanto na pesquisa teórica quando na prática de design. Vamos avaliar aqui os formalimos de apoio a prática de design que apresentamos.

Como partimos de uma perspectiva teórica nova, não identificamos outros formalismos que tenham sido desenvolvidos com o mesmo propósito. Nossos formalismos vieram para preencher a lacuna aberta pelo problema identificado pela Engenharia Semiótica de que designers precisam de ferramentas para expressarem-se melhor. Com este objetivo em mente, nosso trabalho deve ser avaliado pelas contribuições a este aspecto comunicativo do processo de design e pelas características da interface relacionadas com a facilidade de aprendizado.

Nossos formalismos devem ser avaliados como um instrumento que melhora o processo comunicativo - a formulação do conteúdo, da expressão e a interpretação pelos receptores da mensagem - tal como nas outras formas de comunicação como texto, cinema, teatro ou comunicação. Apresentamos uma ferramenta para a formulação do conteúdo (a LEMD) e mostramos como ferramentas para a elaboração da expressão, muito comuns no mercado, podem ser adaptadas para o conteúdo formulado aplicando-se as regras de correlação do SSADIU.

Formalismos e ferramentas de modelagem semântica ou conceitual de sistemas podem também ser utilizadas para formulação do conteúdo. A LEMD tem a vantagem de destacar os elementos essenciais à usabilidade e permitir a inclusão de mensagens de metacomunicação na própria especificação.

Formalismos de especificação do modelo de interação como *gramáticas de ações*, TAGs (Task-Action Grammar), e formalismos baseados em *redes de transição de estados* são complementares à LEMD e podem ser integrados a ele, como discutido no capítulo 5. Em relação a eles, o nosso apresenta a desvantagem de não permitir avaliação do esforço cognitivo de um modelo de interação e seus aspectos de consistência e complexidade. Por outro lado, o SSADIU mostra como as estruturas de um modelo de interação podem ser associadas a widgets e às suas configurações espaciais e temporais, principal problema para a aplicação dos formalismos *textuais* no design de interfaces gráficas.

As regras de correlação, como vimos antes, podem ser consideras regras de design. Entretanto elas não devem ser vistas como *padrões de design* no sentido proposto em [Alexander 77], no qual as regras de design devem ser aplicadas a tipos de problemas no contexto de aplicação. O SSADIU também não pode ser considerado como uma *linguagem de design.* Uma linguagem de design é caracterizada por um *conjunto de elementos, princípios de organização* e *qualificação de situações* [Rheinfrank & Evenson 96]. As regras do SSADIU aplicam-se a um aspecto pontual do design enquanto que o propósito de padrões e linguagens de design é fornecer regras para a elaboração das soluções em potencial aplicadas no domínio de solução. Elas são regras do design de software e não do design de interfaces de usuário. Nosso trabalho está de acordo com esta

perspectiva mais aborda apenas o problema de comunicação que existe no processo de design de interfaces.

Lee apresenta um método de design que mostra como um modelo de objetos podem ser mapeados em widgets das principais ferramentas de interface [Lee 93]. Em seu modelo de objetos ele apresenta categorias de elementos de uma aplicação. Estas categorias são mapeadas nos principais tipos de widgets. A diferença desta proposta para a nossa reside na utilização de um meta-modelo cujos componentes (objetos, recipientes, atributos, ações e outros) são mais adequados à construção de *modelos metafóricos* do que mensagens. Uma limitação é a ausência de um formalismo para a elaboração de modelos de interação com estruturas de articulação mais complexas.

Limitações e trabalhos futuros

Nossa abordagem foi proposta visando complementar os modelos, técnicas e formalismos das abordagens cognitivas. No capítulo 3 procuramos traçar os limites entre estas duas abordagens ressaltando a sua complementaridade. Nossa abordagem apresenta uma linguagem para a especificação da usabilidade visando a sua comunicação através da interface. Ela, no entanto, não determina qual deve ser a melhor estrutura de comandos de acordo com as necessidades e capacidades do usuário. Isso é papel das abordagens cognitivas. A LEMD não deve ser tomada como linguagem de concepção da usabilidade, mas como um formalismo para a especificação abstrata da mensagem do designer. As abordagens da engenharia de software para o design e especificação da funcionalidade são indispensáveis no processo de design de software. Da mesma forma, as técnicas e formalismos para a elaboração dos comandos, visando diminuir o esforço cognitivo (semântico e articulatório), devem ser proporcionadas pelas abordagens cognitivas.

Não abordamos o design da funcionalidade e dos aspectos cognitivos relacionados ao modelo de interação. O primeiro é do escopo de *especificação de software* e o segundo da *engenharia cognitiva* [Norman 86b]. A Engenharia Semiótica precisa ser complementada por estas duas áreas e por *técnicas de avaliação de usabilidade* para responder completamente ao desafio de usabilidade por completo. Reunidas, elas estão na direção de métodos para o *design de software* [Winograd 96].

Nossa abordagem não considerou o design de interfaces multi-usuário que é visto em [Prates 97] e integrado ao nosso em [Prates, Leite & de Souza 98]. Também não abordamos aspectos de manipulação direta em interfaces, já estudadas em [Martin 98] e linguagens de extensão para usuários finais [Silva 98].

O nosso principal objetivo com o SSADIU foi permitir a identificação dos aspectos metacomunciativos a serem considerados no projeto de interfaces e demonstrar que o design deve ser apoiado por um sistema que indique como utilizar signos de interface adequadamente, visando atingir o propósito metacomunicativo. Entretanto, ele precisa ser validado através de uma avaliação que permita contrastar processos de design realizados com e sem o apoio do sistema semiótico às ferramentas convencionais de interfaces GUI/WIMP.

Um processo de avaliação da interpretação da mensagem do designer é fundamental para sabermos se os nossos formalismos atingem o seu objetivo de melhor ensinar o modelo de usabilidade. Este é um ponto que ainda não foi abordado, mas que precisa ser realizado. Esta limitação não compromete o valor do nosso trabalho uma vez que o nosso objetivo foi não apenas produzir boas interfaces, mas apresentar modelos teóricos que solucionassem problemas relacionados à *facilidade de aprendizado*. Os modelos baseados em teoria semiótica apresentam possibilidades para a elaboração de novos formalismos. As distinções conceituais apresentadas por eles são de fundamental importância para se compreender o processo de design e de interação usuário-sistema.

A linguagem de especificação pode incorporar novos conceitos para a modelagem da usabilidade além dos conceitos de signos do domínio, funções aplicadas, visualizações e comandos de funções, permitindo a utilização de conceitos mais específicos do domínio do usuário. O sistema pode ser especificado, por exemplo, em termos de máquinas de softwares, de agentes, recipientes e outros conceitos que também possam ser mapeados em signos. Estes aspectos sistematizam o conceito de metáforas de interface.

Identificamos também a necessidade de desenvolvimento de ferramentas de design de interfaces que incorporem o sistema semiótico. Estas ferramentas devem ter um interpretador da linguagem de especificação. Este interpretador deve apresentar para o

designer quais os widgets mais adequados para cada sentença ou mensagem da especificação, isto é, quais os que comunicam melhor o modelo de usabilidade.

Ferramentas de design automático estão descartadas por diversas razões. O design não é um processo determinístico e a presença humana é fundamental. A linguagem de especificação não determina aspectos de layout da interface uma vez que eles são atividades a serem realizadas por um especialista em design gráfico. Os aspectos estéticos e de layout constituem a base para a elaboração de hipercódigos que podem ser utilizados para incrementar ainda mais o processo comunicativo. As propriedades dos widgets definidas também podem ser manipuladas para comunicar algo ao usuário.

REFERÊNCIAS

ACM SIGCHI 1992. "Curricula for human-computer interaction". Technical report, ACM, NY, 1992. Também em http://www.acm.org/sigchi/.

Adler, P. & Winograd T. *Usability: Turning Technologies into Tools*. Oxford, 1992.

Aho, A.; Sethi, R & Ullman, J. 1986 *Compiler-Principles, Techniques, Tools*. Reading, MA: Addison-Wesley, 1986.

Albus, J.S.; Meystes, A., "A Reference Model Architecture for Design and Implementation of Semiotic Control in Large and Complex Systems", *in Proceedings of 1995 ISIC Workshop*, Monterey, 1995.

Alexander, C. 1977. *A Pattern Language*. New York: Oxford University Press, 1977.

Andersen, P. 1990. *A Theory of Computer Semiotics*. Cambridge University Press, 1990.

Andersen, P.; Holmqvist, B.; Jensen, J.F. (ed.), 1993. *The Computer as Medium*. Cambridge University Press, 1993.

Apple Computer. *The Macintosh Human Interface Guidelines*. Apple Computer, 1992.

Baecker, R. & Buxton, W. "An Historical and Intellectual Perspective", in *Readings in Human-Computer Interaction*, Baecker, R. & Buxton, W. (Eds.) Morgan Kaufmann 1990.

Barfield, L. 1993. *User Interface: Concepts and Design*. Addison-Wesley Publishing Company, 1993.

Berners-Lee, T. Cailliau, R.; Luotonen, A.; Nielsen, H. Secret, A.; "The World Wide web", *Communication of ACM*, 37, 8, 76-82, 1994.

Bodker, S. 1989. "A Human Activity Approach to User Interface". *Human-Computer Interaction*, v4, pp 171-195, 1989.

Booch, G. 1994. *Object Oriented Analysis & Design with Application*. 2.ed. Reading, MA: Bejamim Cummings, 1994.

Booth, P. 1988. *Human Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc. 1988.

Brennan, S. 1991. "Conversation as Direct Manipulation" in B. Laurel *The Art of Human-Computer Interaction*. Reading, MA: Addison-Wesley 1990.

Butler, K. 1996. "Usability Engineering Turns 10", in Interactions, pp. 59-75, jan 1996.

Card, S.; Moran,T. & Newell A. 1983. *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

Carroll, J. *Designing Interaction: Psychology at the Human Computer Interface*. NY: Cambridge University Press, 1991.

Carroll, J (ed) *Scenario-based Design*, 1996.

Carter Jr., J. A. "Combining Task Analysis with Software Engineering in a Methodology for Designing Interactive Systems", in J. Karat (eds.), *Taking Software Design Seriously: Practical Techiniques for Human-Computer Interaction Design*. Boston, MA: Academic Press, 1991.

Casaday, G. "Balance" in J. Karat (eds.), *Taking Software Design Seriously: Practical Techiniques for Human-Computer Interaction Design*. Boston, MA: Academic Press, 1991.

Chomsky, N. *Aspects od the Theory of Syntax* MIT: Cambridge, 1965.

Cordy, J. 1992. "Why the user interface is *not* the programming language - and how it *can* be" in B.Myers (ed.) 1992.

Cowan, D. Ierusalimschy, R.; Lucena, C. & Stepien, T. Abstract Data Views. *Structured Programming*, 14(1):1-13, january 1993.

Curtis, B. & Hefley B. "A WIMP No More: The Maturing of User Interface Engineering", in Interactions, vol. I.1, 22-34, jan 1994.

Dasgupta, S. "The Structure of Design Processes". *Advances in Computers*, v.28,1-67, Academic Press, 1989.

Da Silva, S.; De Souza, C. & Ierusalimschy, R. "A Communicative Approach to End-User Programming Languages", in Lucena, C. (Ed.) *Monografias em Ciência da Computação*, 47/97, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1997.

De Souza, C.S.. "Augmenting Informativeness and Learnability of Items in Large Computer Networks", Technical Report No. CSLI-164. *CSLI, University of Stanford*, 1992.

De Souza, C.S. "The Semiotic Engineering of User Interface Languages". International Journal of Man-Machine Studies 39. Academic Press. 1993. pp. 753-773.

De Souza, C.S. "The Semiotic Engineering of concreteness and abstractness: from user interface languages to end-user programming languages", in. C.J.P. Lucena (ed.) MCC 08/1996, Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil: 1996.

De Souza, C.S.. "Supporting End-User Programming with Explanatory Discourse". In *Proceedings of Intelligent Systems and Semiotics*, Maryland, USA,1997.

De Souza, C.S. & Umiker-Sebeok, J. "Semiotic Engineering in End-User Programming Environments", 1998.

Dix, A.; Finlay, J; Abowd, G & Beale, R. 1993. *Human-Computer Interaction*. Prentice-Hall International, 1993.

Draper, S. Display Managers as the Basis for User-Machine Comunication. in D. Norman & S. Draper (eds.) *User Centered System Design.* Hillsdale, NJ. Lawrence Erlbaum. 1986. Pp.339-352.

Eason, K,; "Towards the experimental study of usability". *Behaviour and Information Technology*, 8 (2), 133-143, 1984.

Eberts, R.. *User Interface Design.* Prentice Hall International Editions. Englewood Cliffs, NJ:1994.

Eco, U.. *A Theory of Semiotics.* Bloomington, IN. Indiana University Press. 1976. Edição brasileira: *Tratado Geral de Semiótica,* coleção estudos 73, ed. Perspectiva, 2a. Edição, São Paulo 1991.

Eco, U.. *Semiótica e filosofia da Linguagem.* Edição brasileira: ed. Ática, São Paulo, 1991.

Eco, U. *Os Limites da Interpretação.* Edição brasileira: coleção estudos 135, ed. Perspectiva, São Paulo, 1995.

Engelbart, D. "A conceptual framework for the augmentation of man's intellect", in Irene Greif (ed.), *Computer Supported Cooperative Work.* Morgan Kauffman, 1988.

Foley, J. & van Dam, A. *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesley, 1982.

Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. "Design Patterns: Abstraction and Reuse of Object-Oriented Design". ECOOP'93 Conference Proceedings, Springer-Verlag, 1993.

Gardner. H. *The Mind New's Science.* 1985. (Tradução brasileira: Edusp, São Paulo, 1995).

Gelernter, D. & Jagannatham, S. 1990. *Programming Linguistics.* Cambridge, MA: The MIT Press, 1990.

Gezzi, C; Jazayeri, M. *Fundamentals of Software Engineering.* Prentice-Hall, 1991.

Gibson, J. *The Ecological Approach to Visual Perception.* Boston: Houghton Mifflin: 1979.

Hartson, H. & Hix, D. "Human-Computer Interface Development: Concepts and Systems for its Management". ACM Computing Surveys, v. 21, no. 1, March 1989.

Hjelmslev, L. *Prolegomena to a Theory of Language*. Menascha: Wincosin University Press, 1963.

Hix, D. & Hartson, H. 1993. *Developing User Interfaces: Ensuring Usability Through Product & Process.* John Wisley & Sons, 1993.

Hooper, C. "Architectural Design: An analogy" . in D. Norman & S. Draper (eds.) *User Centered System Design.* Hillsdale, NJ. Lawrence Erlbaum. 1986.

Howes, A. & Payne, S. "Display-based competence: torwards user models for menu driven interfaces", *in International Journal of Man-Machine Studies*, 33,637-655,1990.

Howes, A. & Young, R. “Learning Consistent, Interactive and Meaningful Task-Action Mappings: A Computational Model”, in *Cognitive Science*, 20, 301-356, 1996.

Hutchins, E.; Hollan, J. & Norman, D. 86. Direct Manipulation Interfaces. in D. Norman & S. Draper (eds.) *User Centered System Design*. Hillsdale, NJ. Lawrence Erlbaum. 1986. pp.87-124

Kammersgaard 1988. “Four differents perspectives on Human-Computer Interaction.” *in International Journal of Man-Machine Studies*, 28, 343-362, 1988.

Kay, A. & Goldberg. “Personal Dynamic Media”, *IEEE Computer* 10 (3), 31-42, 1977

Kieras, D. & Polson, P. “An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 22, 365-394, 1985.

Kieras, D. & Bovair, S. “The Role of Mental models in learning to operate a device”, in Preece, J & Keller, L. *Human Computer Interacion* Pretince-Hall, 1990.

Kitajima, M. & Polson, P. “A comprehension-based model of correct performance and erros in skilled, display-based, human-computer intereaction”, in *International Journal of Human-Computer Studies*, 43, 65-99, 1995.

Jones, C. *Systematic Software Development with VDM*. Prentice-Hall International Series in Computar Sciences, Cambridge: 1986.

Jorna, R. & van Heusden, B. “Semiotics of the user interface”, in *Semiotica* 109 - 3/4, 237-250, 1996.

Lakoff, George *Women, Fire and Dangerous Things: What Categories reveal about the Mind*. University of Chicago Press, Chicago, 1987.

Landauer, T. “Relations Between Cognitive Psychology and Computer System Design”, in J. Carroll (ed.) *Interfacing Thought*. MIT Press, 1988.

Larkin, J. & Simon, H. “Why a diagram is (sometimes) worth then 10000 words”. *Cognitive Science*, 15, 1987.

Laurel, B. *Computers as theatre*, Reading, MA: Addison-Wesley, 1991.

Lee, G. *Object-Oriented GUI Application Development*. NJ: Prentice Hall, 1993.

Lee, A.; Foltz, P. & Polson, P.; “Memory for task-action mappings: mnemonics, regularity and consistency”, in *International Journal of Human-Computer Studies*, 40, 771-794, 1994.

Leite, Jair. *A Interação entre Usuários e Sistemas Computacionais em Linguagem Natural Orientada por Menus*. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, Setembro, 1991.

Leite, Julio. & Franco, A. “A Strategy for Conceptual Model Acquisition” *Proc. Of First IEEE Int. Symposium on Requirements Engineering* IEEE Computer Society Press, 1993.

Licklider “Man-computer Sumbiosis”, *IRE Transactions on Human Factors in Eletronics*, 1960 (reproduzido parcialmente em [Preece & Keller 90]).

Liddle D. “Design of the Conceptual Model” in Winograd, T. (Ed.) *Bringing Design to Software*. Addison-Wesley, 1996.

Lindgaard. G. *Usability Testing and System Evaluation* London, UK: Chapman & Hall, 1994.

McLuhan, M. 1964 *Understanding Media*. NY: New American Library, 1964.

Mullet, K. & Sano, D. *Designing Visual Interfaces*. SunSoft Press - Sun MicroSystems Inc. Mountain View. 1995.

Marcus, A. *Graphic Design for Eletronic Documents and User Interfaces*. New York: ACM Press.

Martins, I. H. Um Instrumento de Análise Semiótica para Linguagens Visuais de Interfaces. *Tese de Doutorado*, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1998.

Maybury, M.T. (ed.) *Intelligent Multimedia Interfaces*. The AAAI/The MIT Press. Cambridge, MA: 1995

Microsoft Corporation *The Windows Interface: Guidelines for Software Design*. Microsoft Press, 1995.

Moles, A *Théorie de l’information et perception esthétique*, Paris: Flammarion, 1958.

Moran, T.. “The Command Language Grammars: a represetantion for the user interface of interactive computer systems. *Int. J. Of Man-Machine Studies*, 15, 3-50, 1981.

Myers, B (ed.) *Languages for developing User Interfaces*. 1992.

Myers, B. “User Interface Software Tools”. ACM Trans. on Human-Computer Interaction,v 1 no.1, 1995.

Nadin, M. “Interface Design: a semiotic paradigm”. Semiotica 69, 3/4, 269-302, 1988.

Nake, F. “Human-Computer interaction - signs and interfacing”in *Languages*, 2, 193-2056, 1994.

Nardi, B. *A Small Matter of Programming*. Cambridge, MA: MIT Press, 1993.

Nielsen, J. *Usability Engineering.* Academic Press, 1993.

Norman, D. & Draper, S. *User Centered System Design*. Hillsdale, NJ. Lawrence Erlbaum. 1986

Norman, D. Cognitive Engineering. in D. Norman & S. Draper (eds.) *User Centered System Design*. Hillsdale, NJ. Lawrence Erlbaum. 1986. Pp.31-61.

Norman, D. *The Design of Everyday Things*. New York:Doubleday, 1990.

Norman, D. “Cognitive Artifacts”, in J. Carroll (ed.) *Designing Interactions: The Psychology*

Norman, D. *Things That Makes Us Smart*. Addison-Wesley, 1993.

Payne, S. “Display-based Action at the User Interface”, in International Journal of Man-Machine Studies, 35, 275-289, 1990.

Payne, S. & Green. T. 1986. “Task-Action Grammar: a model of mental representation of task languages. *Human-Comuter Interaction*, 2(2), 93-133, 1986.

Peirce, C.S. 31-58. *Collected Papers (1931-1958).* Edição brasileira: *Semiótica*. São Paulo, Ed. Perspectiva (coleção estudo, n.46) 1977.

Pimenta, M. & Faust, R. 1997. “Eliciting Interactive Systems Requirements in a Language Centred User-Designer Colaboration: a semiotic approach”, in SIGCHI Bulletin, v.29, n.1, 1997.

Poitrenaud, S. “The PROCOPE semantic network: an alternative to action grammars”, in *Int. J. Human-Computer Studies*, 42, 31-69, 1995.

Prates, R., De Souza, C. & Garcia, A. “A Semiotic Framework for Multi-User Interfaces”, in SIGCHI Bulletin.1997.

Prates, R., Leite, J. De Souza, C. “Semiotically Based User Interface Design Languages”, em Proceedings of IHC’98, *Workshop de Fatores Humano em Sistemas Computacionais*, Maringá, PR, 1998.

Preece, J. & Keller, L. eds. *Human-Computer Interaction.* Prentice-Hall Interface. UK, 1990.

Preece, J.; Rogers, Y.; Sharp, H.; Benton, D.; Holland, S. & Carey, T. 94 *Human-Computer Interaction.* Addison-Wesley, 1994.

Pressman, R. *Software Engineering. A Practiotioner’s Approach.* Fourth edition, MacGraw Hill, 1996.

Reisner, P.; “Formal grammar and human factors design of an interactive graphics system”. *IEEE Transaction on Software Engineering*, SE-7(2), 229-240, 1981.

Rheinfrank, J.; Hartman, W. & Wasserman, A. 92. “Design for Usability: Crafting a Strategy for the Design of a New Generation of Xerox Copiers”. In [Adler &Winograd 92] eds. *Usability.* Addison-Wesley,1992.

Rheinfrank, J. & Evenson, S. 1996. “Design Languages” in [Winograd 96] *Briging Design to Software*, New York: Addison-Wesley, 1996.

Santambrogio, Marco & Violi, Patrizia. *Introduction*, in Umberto Eco, Marco Santambrogio and Patrizia Violi, *Meaning and Mental Representation*, Indiana University Press, Bloomington Indiana, 1988.

Saussure, F. De; *Cours de linguistique générale*, Paris: Payout, 1916.

Simon, H. & Newell, A. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

Shackel, B. 1989. “Human Factors and Usability”, in J. Preece & L. Keller, *Human Computer Interaction.* Cambridge, UK: Prentice-Hall, 1990.

Shaw, M. & Garlan, D. *Software Architecture: Perspective on an Emerging Discipline*, Prentice-Hall, New Jersey:1996.

Sowa, John. *Conceptual Structures; Information Processes in Minds and Machines*, Addison-Wesley Publishing company, New York, 1984.

Stamper, R. "Signs organization, norms and information systems. *Proc Third Australian Conference on Information Systems*, 21-55, Wollongong, Australia, 1992.

Suchman, L. *Plan and Situated Actions: The Problems of Human-Machine Communication.* New York, NY: Cambridge University Press, 1987.

Sutcliffe, A.; & McDermott, M.; "Integrating methods of human-computer interface design with structured systems development", in *International Journal of Man-Machine Studies*, 631-653, 1991.

Sutcliffe, A.; Bass, L.; Cockton, G.; Monk, A. & Newman, I. "Methods, Models and Architectures for Graphical User Interface Design", SIGCHI Bullletin, v.27, n.4, October 1995.

Tennant, H. "Menu-based Natural Language Understanding", in AFIPS Conference Proceedings, 1984.

Tufte, E. *Envisioning Information*, Cheshire, CT: Graphics Press,1990.

Weitzman, L. & Wittenburg, K. 1994. "Automatic Presentation of Multimedia Documents Using Relational Grammars". ACM Multimedia 94 Proceedings. 443-451, 1994.

Wasserman, A. "Towards a Discipline of Software Engineering", in *IEEE Software*, 1996.

Wasserman, A.; Pircher, P.; Shewmake, D. & Kersten, M. Developing Interactive Information Systems with User Software Methodology". In R. Baecker & W. Buxton (eds.) *Readings in Human-Computer Interaction: A multi-disciplinary approach.* Los Altos, CA: Morgan Kaufman.

Winograd, T. " From Programming Environments to Environments for Designing". Communications of ACM, v.38, no.6, June 1995.

Winograd, T. (Ed.) *Bringing Design to Software*. Addison-Wesley, 1996.

Winograd, T. & Flores, F. Understanding Computers and Cognition: New foundations for design. 1986.

Wroblewski, D. The Construction of Human-Computer Interfaces Considered as a Craft, in J. Karat (eds.), *Taking Software Design Seriously: Practical Techiniques for Human-Computer Interaction Design.* Boston, MA: Academic Press, 1991.

Young, R.; Green, T. & Simon, T. "Programmable User Models for Predictive Evaluation of Interface Designs", in *Proc. CHI'89*, ACM Press: NY, 1989.

Zemanek, H. "Semiotics and programming languages". Communications of ACM, v.9, no.6, 139-143, 1966.