Uma Linguagem de Especificação para a Engenharia Semiótica de Interfaces de Usuário

Jair Cavalcanti Leite
Dep. de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Campus Universitário, Lagoa Nova
Natal, RN, 59072-970 Brasil
+55 +84 215 3817
jair@dimap.ufrn.br

Clarisse Sieckenius de Souza
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Av. Marquês de São Vicente, 225, Gávea
Rio de Janeiro, RJ, 22453-900 Brasil
+55 +21 512 2299 (r.4344)
clarisse@inf.puc-rio.br

RESUMO
Na abordagem da Engenharia Semiótica para o *design* de interfaces de usuário o sistema é considerado um artefato de metacomunicação através do qual o *designer* envia para os usuários uma mensagem expressa através da interface, cujo conteúdo é o modelo conceitual da aplicação. A mensagem do *designer* tem uma natureza dinâmica e interativa, pois é formada pela conjunto de signos - palavras, gráficos, figuras, sons, etc. - trocados entre o usuário e sistema durante o processo de interação. Este trabalho contribui para a Engenharia Semiótica, apresentando um modelo para a interface como expressão da mensagem do *designer* e um formalismo lingüístico para a sua especificação. Com isto, o *designer* pode especificar a sua mensagem de forma estruturada e abstrata para depois mapeá-la em objetos de interfaces convencionais.

Palavras chave
Design de Interfaces de Usuário, Engenharia Semiótica, Usabilidade, Interação Humano-Computador.

ABSTRACT
In the Semiotic Engineering approach to user interface *design* computer systems are considered as a metacommunication artifact that conveys a message from the *designer* to the user. The *designer*'s message content is the application's conceptual model, and its expression is the user interface itself, which has a dynamic and interactive nature. Our work contributes to Semiotic Engineering by proposing a model for the user interface as the expression of a *designer*'s message, and a linguistic formalism for its specification. With these resources, the *designer* can specify his/her message in a structured and abstract way and map this specification onto conventional user interface objects.

Keywords
User Interface Design, Semiotic Engineering, Human-Computer Interaction, Usability.

1. INTRODUÇÃO

A *usabilidade* é um conceito que se refere à qualidade da interação de sistemas computacionais interativos com os seus usuários e depende de vários aspectos. Diversas abordagens teóricas têm apresentado propostas para o processo de *design* de sistemas que visam aumentar a sua usabilidade. Entretanto, a maioria destas abordagens apresentam como solução a este desafio modelos e técnicas para o *design* pautadas em características cognitivas dos usuários (Norman 1986). Estes modelos cognitivos são insuficientes para o desafio de usabilidade por não articularem o papel do projetista e da interface na aquisição do modelo de usabilidade pelo usuário.

Nosso trabalho adota a perspectiva da Engenharia Semiótica para o *design* de interfaces de usuário na qual sistemas são considerados como artefatos de *metacomunicação* (de Souza 1993). Nesta perspectiva a interface é vista como uma mensagem unidirecional e indireta de *designer*s para usuários. Esta mensagem se caracteriza pela sua capacidade de, ela própria, enviar e receber mensagens durante o processo de interação entre o usuário e o sistema. O aspecto de usabilidade que a Engenharia Semiótica visa resolver é como o conhecimento que o usuário precisa adquirir para utilizar melhor o sistema pode ser melhor "ensinado" através da interface de usuário.

As principais idéias da Engenharia Semiótica que utilizamos em nosso trabalho já foram introduzidas e discutidas em Prates, Leite e de Souza (1998). Neste trabalho nosso objetivo específico é apresentar uma linguagem para a especificação da interface como uma **Mensagem do *Designer*** que comunica para os usuários o modelo de interação e de funcionalidade do sistema, aqui chamado de **Modelo de Usabilidade**. O modelo de funcionalidade determina o que o usuário pode fazer enquanto o modelo de interação determina como ele pode utilizar o sistema.

A Linguagem de Especificação da Mensagem do *Designer* (LEMD) contribui para a Engenharia Semiótica possibilitando uma especificação da interface em nível abstrato, que permite a representação de associações entre

intenções de comunicação do *designer* e expressões de interface utilizadas para este fim, como, por exemplo, os *widgets* presentes nas ferramentas de desenvolvimento de GUIs, detalhadamente apresentado em trabalho anterior (Leite 1998).

Apresentamos o perfil geral da LEMD e um exemplo de como a linguagem é capaz de representar os diversos tipos de mensagens que podem ser veiculadas na interface de usuário. Na conclusão, discutimos as vantagens de se utilizar a LEMD, na medida em que ela reflete um modelo global de comunicação humana através de artefatos de metacomunicação (as interfaces de sistema) e com isto motiva o *designer* de interface a considerar não somente os processos interpretativos do usuário (como extensamente explorado em abordagens cognitivas), mas principalmente os processos comunicativos e expressivos que ele pode utilizar com maior consciência e intencionalidade para atingir metas comuns a todos os que visam aumentar a usabilidade de aplicações computacionais.

2. A MENSAGEM DO *DESIGNER*

Na Engenharia Semiótica o foco está na comunicação interpessoal entre o *designer* e os usuários. O que o *designer* transmite não é uma mensagem como a de um documento multimídia, um livro, ou um filme, mas uma mensagem interativa e dinâmica: 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 de maneira a revelar para o usuário o espectro de usabilidade da aplicação. O *designer* 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!"*. O usuário tem um duplo papel: o de agente da interação e o de receptor na comunicação com o projetista.

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. O conceito de signo como apresentado por Peirce (1931) mostra-nos que a mensagem que o *designer* envia para os usuários têm como expressão a interface de usuário e como conteúdo a funcionalidade e o modelo de interação definidos pelo programa que implementa o sistema. O interpretante deste signo é, para o usuário, o modelo conceitual que ele adquire a partir da interpretação da interface - que é a expressão da mensagem - durante o processo de interação.

Por exemplo, o usuário, quando observa um *widget* na interface, precisa saber como pode utilizá-lo e qual será o comportamento do sistema após a sua ação. O que está sendo interpretado pelo usuário, mesmo sem perceber, 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"*. 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.

Para ilustrar os conceitos da nossa abordagem e exemplificar a utilização da Linguagem de Especificação da Mensagem do Designer vamos apresentar o exemplo da especificação de parte de uma interface para o comando imprimir de uma aplicação qualquer. A Figura 1 mostra a interface para este comando que o *designer* construiu para expressar o que o usuário pode fazer e como ele pode fazer. Ao longo do artigo vamos fazer as referências a esta interface como mensagem do *designer*.

Podemos dizer que informalmente esta mensagem expressa a solução do *designer* para que o usuário controle a função de impressão de arquivos. Ele deve fornecer o nome do arquivo, escolher a impressora desejada e indicar quantas cópias ele deseja. Após fornecer estas informações o usuário deve escolher quais as opções de controle o *designer* disponibilizou para esta função – neste exemplo elas são *iniciar*, *terminar*, *suspender*, *continuar* e *cancelar*.

Figura 1: A interface para o comando imprimir.

Quase tudo na interface tem o potencial de ser signo. Na figura 1, as palavras, os *widgets*, as bordas e linhas funcionam como signos para o usuário. Estes signos são, na maioria das vezes, provenientes de diversos códigos. Para que o *designer* consiga elaborar a expressão que permita que o usuário adquira um modelo de usabilidade de acordo com o que ele projetou é preciso um processo de

produção de signos de interface apoiado por um sistema semiótico.

Projetar interfaces é, portanto, projetar uma mensagem complexa, interativa e unidirecional, destinada a usuários de aplicações computacionais. 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.

A Teoria da Produção de Signos de Eco (1976) nos mostra que é preciso estruturar o sistema expressivo e o sistema semântico e mapear seus componentes através de regras de mapeamento semântico. A proposta de Engenharia Semiótica original (de Souza 1993) 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 uma linguagem para especificação que pudesse ser integrada a um processo de *design*, combinando conceitos da teoria dos códigos e da produção de signos. Os conceitos destas teorias determinaram como requisitos a necessidade de se fazer uma estruturação da interface e do modelo de usabilidade que são, respectivamente, a expressão e o conteúdo da mensagem do *designer*.

Para que a interface possa ser utilizada como expressão da mensagem do *designer* vamos apresentar um modelo baseado nos tipos mais comuns de interfaces gráficas. Neste modelo a interface é formada pelo ***medium* interface**, pela **ferramenta de acionamento** e por **signos de interface**. Para as interfaces gráficas, como é o caso da figura 1, o *medium* é a tela da interface. As ferramentas de acionamento são mouse com o qual o usuário pressiona os objetos e o teclado com o qual ele fornece valores alfanuméricos. Os signos de interface são os elementos fundamentais que devem ser elaborados pelo usuário de maneira que o usuário adquira o modelo concebido pelo *designer*. Embora qualquer distinção possa funcionar como signo, nas interfaces gráficas os *widgets* são signos de interfaces pré-definidos e devem ser explorados pelo *designer*. Na figura 1, os *botões de acionamento* são utilizados para o usuário acionar opções de controle, *caixas de texto* são utilizados para o fornecimento de dados e *botões de opção* para permitir a escolha de valores específicos.

O modelo para a interface como expressão permite ao *designer* construir uma expressão através de signos de interface. Os signos de interface são uma abstração do que pode ser expresso através da interface e quais os seus potenciais significados. Eles são também os responsáveis pela comunicação usuário-sistema. Estes signos são dispostos espacial e temporalmente no *medium* interface, que no caso das interfaces convencionais é a tela do computador.

A estruturação do conteúdo da mensagem é motivada pelo modelo do processo de interação apresentado por Norman (1986) que descreve os passos necessários para o usuário realizar o *mapeamento tarefa-ação*. A nossa escolha justifica-se pela ampla divulgação e aceitação que este modelo teve entre pesquisadores da área. Baseado no modelo de Norman, podemos dizer que o modelo conceitual da aplicação deve conter **funções da aplicação** que o usuário possa utilizar para atingir as metas estabelecidas. Estas funções atuam sobre signos que representam conceitos do domínio e são chamado de **signos do domínio**. As funções da aplicação são controladas por uma estrutura de ações que compõem um **comando de função**. A ativação de uma função pelo seu comando causa uma mudança no estado dos signos do domínio. Os signos do domínio, as funções da aplicação e os comandos de funções são os tipos de elementos básicos que formam o conteúdo da mensagem e precisam ser expressos através dos signos de interface.

No exemplo da figura 1, temos uma única função da aplicação – a função *impressão*. São signos do domínio *arquivos, cópias e impressoras*. A interface apresenta para o *designer* o comando associado a esta função e indica quais são as informações sobre os signos do domínio e como o usuário pode acionar os controles da função. O nosso modelo diferencia o comando da função dos controles específicos de sua execução. Para cada comando o *designer* deve possibilitar que o usuário controle o início, a parada, o cancelamento, a interrupção e a continuação.

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 da aplicação, tarefas, ações, estados do sistema.

Este modelo distingue o que é essencial para a utilização da aplicação daquilo que é metacomunicação que o *designer* envia para o usuário com o objetivo de melhorar a usabilidade do sistema. Grande parte dos elementos da interfaceestão a serviço da metacomunicação. Uma pequena parte dos *widgets* é que é essencial para a utilização do sistema. A interação que o usuário realiza apenas com a interface é chamada de leitura interativa da mensagem do *designer* e faz parte do processo de metacomunicação. Um exemplo disto é a navegação através de menus e janelas que o usuário precisa realizar para acionar um comando. Alguns dos acionamentos com os menus ou com os botões de controle de janelas não são

essenciais para a interação. Apenas o *widget* correspondente ao comando é essencial. Contudo, todos eles exercem importância para a usabilidade.

3. A ENGENHARIA SEMIÓTICA DE INTERFACES DE USUÁRIO

Na abordagem da Engenharia Semiótica, o desenvolvimento da interface de usuário ocorre num processo formado por quatro etapas básicas: *análise, design, prototipação e avaliação*. Estas três últimas atividades ocorrem num processo cíclico no qual o *design* da interface pode ser corrigido ou evoluir para novas etapas a partir da avaliação da interface, como mostra a Figura 2. O processo de design é visto como um processo de produção de mensagem e deve ser conduzida em duas etapas: a formulação do conteúdo e a elaboração da expressão da mensagem do *designer* para o usuário. A formulação do conteúdo é realizada com o apoio da linguagem de especificação que apresentamos neste trabalho. A elaboração da expressão é feita através da construção do protótipo. O resultado deste processo iterativo pode ser diversos protótipos de interfaces que expressem diversas mensagens alternativas.

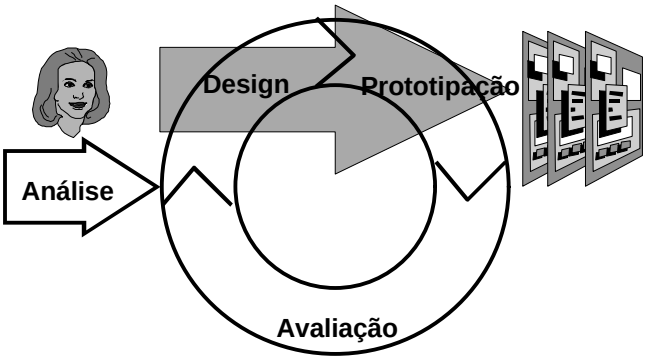

Figura 2: O processo de design na Engenharia Semiótica.

Na formulação do conteúdo, o *designer* deve conceber e especificar os componentes do modelo de usabilidade. Ele deve especificar os signos do domínio e as funções da aplicação e as estruturas dos comandos e as visualizações de funções da aplicação e signos do domínio. Existem diversos métodos e linguagens para a especificação da funcionalidade e do modelo de interação de um sistema. Linguagens de especificação funcional podem ser utilizadas para a especificação dos componentes do modelo de funcionalidade, enquanto que o modelo de interação pode ser especificado por linguagens ou formalismos de especificação da interação.

O *designer* precisa complementar esta especificação considerando que estes componentes são parte do processo comunicativo existente entre o *designer* e o usuário. Ele deve realizar a especificação de cada componente do modelo de usabilidade como parte da mensagem global que é enviada ao usuário. Esta especificação deve ser abstrata uma vez que a mensagem é descrita independentemente de como será a sua expressão final, ou seja, de quais serão os *widgets* utilizados para expressá-la. Usando a Linguagem de Especificação da Mensagem do *Designer* (LEMD) é possível descrever a mensagem de maneira abstrata e estruturada. A especificação da mensagem do *designer* inclui a descrição da funções da aplicação, dos signos do domínio e dos comandos sem a preocupação com quais *widgets* e configurações espaciais e temporais elas serão expressas. Apresentamos esta linguagem na próxima seção.

Por fim, a expressão concreta da mensagem deve ser concebida e especificada para que possa ser apresentada ao usuário durante o processo de interação. O nosso modelo para a interface como expressão da mensagem considera que os signos de interfaces são veiculados através de um *medium* (como, por exemplo, a tela da interface) e articulados em configurações espaciais e temporais. Os signos de interface 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 (conteúdo da mensagem) são os elementos do modelo de usabilidade. Isto significa que o *design* da interface de usuário deve estar associado semanticamente com a especificação da funcionalidade e do modelo de interação do sistema. Esta associação semântica é facilitada através da especificação abstrata da mensagem usando a LEMD.

Resumindo, na Engenharia Semiótica o *design* é apoiado pela LEMD que oferece a possibilidade de construir os diversos tipos de mensagens que possam ser conduzidas pela interface de usuário. Estas mensagens são abstrações dos principais *widgets* disponíveis nas ferramentas de desenvolvimento de interfaces. A LEMD funciona como sistema semântico que permite descrever estes componentes do conteúdo da mensagem. Isto significa que cada elemento do modelo de usabilidade que tenha sido concebido e especificado deve estar associado por regras de correlação semântica a um signo de interface.

A Figura 3 mostra o processo de *design* de interfaces compreendendo as etapas descritas acima e os principais formalismos e ferramentas de apoio a este processo. A especificação do modelo de usabilidade pode ser feita apoiada pelas linguagens e formalismos tradicionais de especificação funcional e de especificação do modelo de interação. A especificação abstrata da mensagem é feita com o auxílio da LEMD cujas estruturas são mapeadas em tipos de signos de interface (*widgets*) das principais ferramentas de *design*.

4. A LINGUAGEM DE ESPECIFICAÇÃO DA MENSAGEM DO *DESIGNER*

A Linguagem de Especificação da Mensagem do *Designer* (LEMD) tem como objetivo apoiar a formulação da mensagem sobre o modelo de usabilidade. Ela permite

especificar os *signos do domínio, as funções da aplicação* e *os comandos de função* como sendo mensagens enviadas pelo *designer*. As sentenças da linguagem descrevem as *mensagens de interação* que compõem a mensagem global. Estas mensagens são descritas de maneira abstrata independentes de quais signos de interfaces serão utilizados.

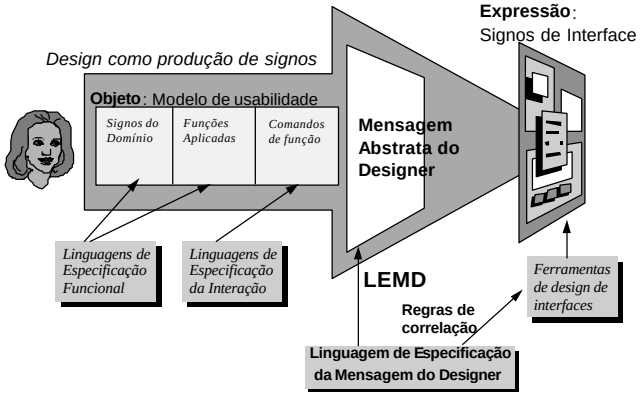

Figura 3: Formalismos de apoio ao *design*

4.1 A estrutura da mensagem do *designer*

Seguindo o modelo para o conteúdo da mensagem, a linguagem deve permitir ao *designer* especificar os signos do domínio, funções de aplicação, e comandos. Além disso, o *designer* deve utilizar diversas outras mensagem que comuniquem ao usuário o conjunto de tarefas que ele pode realizar, o processo de navegação através das telas da interface e informações diretas que ajudem ao usuário interagir como o sistema. Este apoio à realização de tarefas ocorre através do envio de mensagens de metacomunicação diretas e indiretas. Esta estrutura básica permite diferenciar entre os signos de interface que são essenciais para o usuário utilizar o sistema e as mensagens que organizam estes signos ou que auxiliam o usuário a aprender sobre ele.

A LEMD diferencia diversos tipos de mensagens. A figura 4 retoma o exemplo mostrado na figura 1 identificando alguns deste tipos de mensagens do designer.

As **mensagens sobre estados de signos do domínio** revelam o estado do sistema e permitem ao usuário avaliar se a sua meta foi atingida. O estado de um signo como "nome do arquivo" deve ser representado adequadamente para os usuários. O *designer* deve comunicar adequadamente este signo através de tabelas de números ou de gráficos, por exemplo. Na figura 4, o *designer* utiliza texto para representar o signos do domínio nome do arquivo, nome da impressora e número para representar número de cópias.

As **mensagens sobre funções da aplicação** revelam o seu estado operacional e o que o usuário deve fazer para controlá-la. Uma das grandes deficiências das aplicações atuais é falta de representação e controle operacional de funções. Na maioria das vezes o único retorno que o usuário tem é o resultado final. O *designer* deve poder mostrar o desempenho do sistema e permitir que o usuário interrompa ou reinicie quando a função estiver sendo executada. Não mostramos mensagens sobre o estado operacional da função impressão. Entretanto, um signo que mostre se a função está imprimindo ou se houve alguma interrupção é um exemplo de mensagem deste tipo.

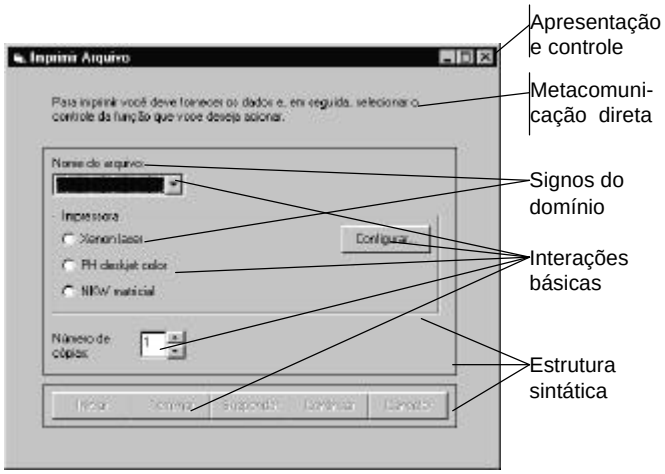

Figura 4: As mensagens do *designer*

As **mensagens sobre a estrutura sintática dos comandos** revelam a estrutura e a articulação das interações que o usuário precisa desempenhar. A estrutura sintática determina como as interações básicas podem ser articuladas na formação de comandos compostos. As interações podem ser agrupadas em seqüência (**Sequence**), repetição (**Repeat**), agrupamento (**Join**), combinação (**Combine**) e seleção (**Select**). Na interface para o comando impressão utilizamos algumas destas estruturas. Por exemplo, para que o usuário possa fazer o controle operacional da função ele deve primeiro fornecer as informações associadas aos signos do domínio. Neste caso, existe uma estrutura seqüencial expressa por um layout vertical com dois quadros distintos e pelas cores cinza dos botões de acionamento, dando a idéia de que eles não estão disponíveis, até que o usuário forneça as informações no quadro superior. Com estas estruturas podemos especificar comandos de inúmeras interfaces gráficas.

As **mensagens sobre interações básicas** indicam ao usuário a interação a ser desempenhada. As interações básicas previstas na linguagem são *acionar* (**Activate**), *fornecer informação* (**Enter**) e *selecionar informação* (**Select**). O acionamento pode ser comunicado através de *botões de acionamento*. O fornecimento de informações, expresso na LEMD através de **Enter** pode ser expresso por diversos *widgets*. Para o usuário fornecer as informações textuais no exemplo da *impressão* o *designer* optou por utilizar *caixa de texto*. Para as escolhas ele optou por

botões de opção e para valores numéricos uma *caixa numérica com botões de incremento*.

As **mensagens de metacomunicação de assistência a tarefas** auxiliam ao usuário a realizar tarefas compostas por mais de um comando. Os assistentes de tarefas (*wizards*) que auxiliam na instalação de programas são exemplos deste tipo de mensagem.

As **mensagens de metacomunicação para apresentação e controle da leitura da mensagem** comunicam como o usuário deve "ler" a própria mensagem do *designer*. A navegação entre telas, a ação de mover, aumentar e diminuir janelas são exemplos de ações que o usuário faz para "ler" a interface, como quem folheia um livro. Estas ações do usuário não modificam o estado funcional do sistema e, portanto, não são consideradas comandos de função em nosso modelo. A LEMD diferencia os controles de leitura dos comandos convencionais chamando a atenção do *designer* para este aspecto. Quando o usuário aciona o botão de acionamento "Configurar..." na figura 4, nenhum controle operacional da função impressão é acionado. O *designer* tem a intenção de comunicar que ao pressionar este botão o usuário quer ativar um outro comando de função que permitirá configurar a impressora. Embora não esteja mostrado na figura, uma nova janela deve aparecer para mostrar como o usuário deve proceder para comandar esta função.

As **mensagens de metacomunicação direta**, que permitem ao *designer* enviar uma mensagem diretamente ao usuário para se referir a qualquer outro elemento da interface, inclusive à própria mensagem. Estas mensagens são especificadas através do elemento **View** da LEMD. Na figura 4, o texto que vem na parte superior da janela "Para imprimir você deve..." é uma mensagem do *designer* que apresenta a função e as ações que o *designer* deve desempenhar para comandá-la.

Após a especificação da mensagem o *designer* pode mapear sentenças da linguagem em *widgets* ou combinações de *widgets* utilizando as **regras de mapeamento semântico**. Estas regras mostram, por exemplo, que **Activate** pode ser mapeado em *botões de acionamento*. Isto permite a concretização da especificação abstrata em protótipos da interface para que possam ser avaliados a partir de testes de usabilidade.

4.2 Especificação de Signos do Domínio

A Engenharia Semiótica requer que a funcionalidade seja especificada em termos de objetos que representam informações de conceitos do domínio. A LEMD permite 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 especifica signos do domínio possui a seguinte forma geral[1]:

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

As sentenças abaixo descrevem os signos do domínio para o exemplo das figuras 1 e 4:

```
Domain-Sign Número de cópias
   representation-type number
Domain-Sign Nome do arquivo representation-type name
Domain-Sign Nome da Impressora
   representation-type finite set-of-name
```

Os tipos de representação de signos do domínio são descritos de forma abstrata e podem ser numéricos, nomes, conjuntos, listas e vários outros comuns a maioria das linguagens de especificação tradicionais ou das linguagens de programação.

4.3 Especificação de Funções da Aplicação

As funções da aplicação 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 da aplicação e devem ser utilizadas na especificação do comando.

A especificação de uma função da aplicação 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 são descritas aqui usando uma linguagem informal. Entretanto, elas devem ser descritas por uma linguagem de especificação mais apropriadas, como as linguagens formais. A função da aplicação *Impressão* da figura 1, por exemplo, pode ser especificada como na Figura 5.

```
Application-Function Impressão
  Operands
```

[1]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`.

```
      Nome do arquivo,
      Nome da Impressora,
      Número de cópias
   Pre-conditions
      Nome do arquivo deve ser definido,
      Nome da impressora deve ser
                  escolhida,
      Número de cópias deve ser fornecido,
           caso contrário imprime apenas
           uma cópia
   Post-conditions
      A impressora especificada deve
           produzir o número de cópias
           fornecidas do arquivo definido.
   Control   Iniciar, Parar, Suspender,
             Continuar, Cancelar
   State      Disponível, Executando,
             Interrompido, Terminado
```

Figura 5: Especificação da função *imprimir*

4.4 A especificação para a mensagem para comando *imprimir*

Suponhamos que o *designer* tenha a intenção de desenhar a parte da interface para o comando de função imprimir de uma aplicação com as seguintes características. O comando requer que o usuário forneça o nome do arquivo a ser impresso, o nome da impressora e o número de cópias. O usuário deve poder escolher dentre as impressoras disponíveis e também configurá-la. A função da aplicação *impressão* a ser controlada permite os controles *iniciar*, *terminar*, *suspender e continuar*.

```
Command-Message Imprimir for Application-Function
Impressão
 Join {
   View "Para imprimir você deve fornecer os
         dados, e, em seguida, selecionar o
         controle da função que você deseja
         acionar."
   Sequence {
     Join {
       Select {
          Enter Information-of Nome do Arquivo
          Select Information-of Nome do Arquivo
       }
       Combine {
         Select Information-of Nome da impressora
         Activate Show Command-Message Configurar
           impressora
       }
       Enter  Information-of Número de cópias
   }
   Select {
      Activate  Start          Application-Function
        Impressão
      Activate  Stop           Application-Function
        Impressão
      Activate   Suspend       Application-Function
        Impressão
      Activate    Continue     Application-Function
        Impressão
      Activate  Waive          Application-Function
        Impressão
    }
  }
}
```

Figura 6: Especificação da mensagem para comando da função *imprimir*

O *designer* decidiu que o usuário apenas pode realizar os comandos de controle após ter fornecidos os dados. Neste caso os grupos de interações para fornecimentos dos dados e controle operacional devem estar estruturados com uma seqüência (**sequence**). Para auxiliar o usuário ele optou por enviar uma mensagem direta informando a respeito da estrutura seqüencial através de uma mensagem **view**. O fornecimento dos dados pode ser realizado em qualquer ordem (**join**). Para fornecer o nome do arquivo o usuário pode escolher entre digitar o nome diretamente ou selecionar de uma lista de nomes de arquivos. Para configurar uma impressora é necessário escolhê-la para ativar a mensagem de comando *configurar impressora*. Esta dependência entre as duas ações do usuário requer a utilização da estrutura combinação (**combine**). O número de cópias deve ser fornecido diretamente. Após ter fornecido os dados, o usuário pode escolher o controle operacional da função desejado.

Esta especificação do modelo de interação pode ser descrita como uma mensagem utilizando a LEMD. A especificação da mensagem que comunica este modelo está ilustrada na Figura 6.

Esta mensagem deve ser expressa através dos *widgets* de uma interface gráfica. As construções da interface estão associadas a regras de correlação de acordo com a Tabela 1.

Join	Configuração espacial (agrupamento visual)
Sequence	Configuração espacial com os elementos da seqüência em cinza claro indicando "não disponível"
Combine	*widgets* Frame
Select	Configuração espacial (agrupamento visual)
Select texto	Caixa "Combo"
Enter texto	Caixa de texto
Enter numérico	Caixa Spin (número com incrementadores)
Activate	Botão de Pressão

Tabela 1: Regras de mapeamento semântico utilizadas.

A partir da especificação da mensagem do *designer* e aplicando as regras de correlação associadas a signos de interface do MS Visual Basic 5, pode-se elaborar a mensagem final mostrada na Figura 1.

5. CONCLUSÕES

As abordagens cognitivas para o *design* de interface normalmente buscam modelos do funcionamento da mente humana tentando entender quais características as

interfaces devem ter para que elas sejam mais fáceis de usar e de aprender. Estas abordagens são de fundamental importância para a elaboração do modelo conceitual da aplicação. Elas, entretanto, deixam de considerar o amplo potencial comunicativo que as interfaces têm e que podem ser explorados no aprendizado deste modelo. A Engenharia Semiótica tem como objetivo explorar esta característica de metacomunicação que os sistemas computacionais possuem e oferecer ao *designer* instrumentação que o permita *ensinar* quais soluções ele projetou para os problemas dos usuários.

Nosso trabalho aplica conceitos teóricos de semiótica que explicam fenômenos de natureza comunicativa que contribuem para o entendimento da interação humano-computador e para o *design* de interfaces de usuário. Na Engenharia Semiótica o processo de *design* é visto como uma atividade semiótica, isto é, que inclui processos de produção de signos. 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 uma linguagem de especificação – a LEMD – de maneira que se tenha uma ferramenta que auxilie o *designer* neste processo comunicativo.

A LEMD permite estruturar os diversos tipos de mensagens que o *designer* necessita enviar para comunicar para o usuário o que ele pode fazer (as funções de aplicação que operam sobre os signos do domínio) e como ele pode utilizar o sistema (os comando de função). Além disso, a LEMD possibilita ao *designer* especificar as diversas outras mensagens auxiliam o processo de utilização e aprendizado do sistema.

Utilizando a LEMD as mensagens são especificadas de maneira abstrata independente de quais signos de interface serão utilizados para concretizá-las. Regras de mapeamento semântico mostram como a especificação abstrata da mensagem pode ser mapeada em *widgets* de ferramentas de interfaces gráficas. A LEMD também permite analisar as intenções comunicativas do *designer* e pode ser a base para avaliação de interfaces existentes.

Os beneficiários de nosso trabalho são pesquisadores de IHC, *designer*s e usuários. Os pesquisadores dispõem de modelos teóricos com explicações sobre fenômenos da interação usuário-sistema que apresentam perspectivas complementares 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 tais fenômenos como processos semióticos (Peirce 1931). Os pesquisadores dispõem ainda de modelos dos componentes de funcionalidade e de interatividade que podem ser utilizados na construção de outras ferramentas.

Os *designer*s também são beneficiários diretos por utilizarem um formalismo complementar àqueles que normalmente dispõem no *design* da interface. A LEMD permite-lhes formularem sua mensagem de maneira estruturada e coerente com o modelo apresentado. Este modelo descreve os componentes necessários ao desempenho do processo de interação. Este formalismo é complementar às linguagens de especificação tradicionais e pode ser integrado às ferramentas de interfaces baseadas em *widgets*, normalmente utilizadas para interfaces gráficas.

O usuário é um beneficiário indireto quando utiliza sistemas cujas interfaces produzidas com o apoio da LEMD que são mais fáceis de interpretar. Ele 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.

A linguagem proposta não foi comparada com outras linguagens uma vez que ela faz parte de uma abordagem inovadora que é complementar às abordagens cognitivas mais tradicionais. A especificação da mensagem utilizando a LEMD, não garante sozinha que o sistema construído tenha boa usabilidade. Ela deve ser utilizada conjuntamente com outros formalismos tradicionais da engenharia de software que façam a especificação da funcionalidade e do modelo de interação. O seu objetivo é especificar como estes componentes do modelo conceitual de aplicação deve ser comunicados aos usuários.

Nosso próximo passo visa integrar esta linguagem com métodos de *design* e engenharia de software para que possamos avaliar a sua efetiva contribuição no processo de *design*. Também estamos iniciando testes que permitam avaliar o impacto da LEMD na usabilidade e na comunicabilidade das interfaces produzidas.

6. REFERÊNCIAS

De Souza, C.S. “The Semiotic Engineering of User Interface Languages”. International Journal of Man-Machine Studies 39. Academic Press. 1993. pp. 753-773.

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.

Leite, J. C. *Modelos e Formalismos para a Engenharia Semiótica de Interfaces de Usuário*. Tese de Doutorado. Departamento de Informática. PUC-Rio, 1998.

Norman, D. Cognitive Engineering. in D. Norman & S. Draper (eds.) User Centered System *Design*. Hillsdale, NJ. Lawrence Erlbaum. 1986. Pp.31-61.

Peirce, C.S. 31-58. *Collected Papers (1931-1958)*. Edição brasileira: *Semiótica*. São Paulo, Ed. Perspectiva (coleção estudo, n.46) 1977.

Prates, R., Leite, J. & de Souza, C. “Semiotically-based User Interface Representation Languages”. Atas do *I Workshop de Fatores Humano em Sistemas Computacionais*, Maringá, PR, 1998.