3
Conceitos e Representações de *Design* de IHC à luz da Engenharia Semiótica

Este capítulo descreve conceitos e representações de *design* necessários ao entendimento e uso do modelo de interação proposto. Na seção 3.1 serão apresentados alguns conceitos da Engenharia Semiótica que são fundamentais na construção das representações de *design*. Na seção 3.2 serão apresentados os cenários de uso, estendidos de forma a motivar a reflexão dos *designers* sobre as tarefas que o sistema irá apoiar. Na seção 3.3 será descrita uma adaptação de um modelo de tarefas existente para contemplar conceitos da Engenharia Semiótica e elementos que servirão de insumo para o modelo de interação proposto.

3.1 Conceitos da Engenharia Semiótica

Dois conceitos da Engenharia Semiótica que são fundamentais para a concepção e uso de representações de *design* baseadas nesta teoria são signos e rupturas na comunicação preposto do *designer*–usuário. Estes conceitos serão descritos a seguir.

Signos

A Engenharia Semiótica herdou da Semiótica (Peirce, 1931-1958) a noção de signo como "qualquer coisa que signifique algo para alguém". Esta noção de signos constitui uma base para a ontologia[6] da aplicação e por isto deve ser utilizada no processo de *design.* Os signos provêem um fio condutor para o

[6] Ontologia é o termo utilizado para se referir ao conhecimento compartilhado de algum domínio de interesse. Ela pode ser utilizada como um *framework* único para se resolver os problemas, de uma determinada forma. Uma ontologia freqüentemente contém um conjunto de conceitos (por exemplo, entidades, atributos e processos), suas definições e relações (Uschold e Gruninger, 1996).

discurso do *designer*, desde as representações de *design* até se manifestarem na interface (Figura 3.1). Estes constituem o vocabulário a ser compartilhado não apenas entre o preposto do *designer* e os usuários, mas também entre os membros da equipe de *design*.

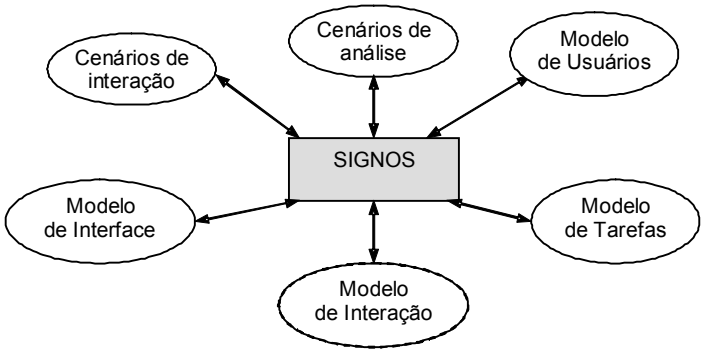

Figura 3.1: A utilização de signos como base das representações do processo de *design*.

Retomando o projeto do Quadro de Avisos, um signo utilizado do começo ao fim foi, por exemplo, o "texto do aviso". Ele foi detectado na fase de análise através de entrevistas com os usuários, e esteve presente em todas as demais fases e representações de *design*, sempre com a denominação de "texto do aviso", até ser incluído como tal na interface do produto final.

Rupturas na Comunicação Preposto do *Designer*–Usuário

Assim como ocorrem mal-entendidos nas conversas entre pessoas, a Engenharia Semiótica reconhece que falhas de comunicação são parte inerente da conversação usuário-preposto do *designer*. Estas falhas incluem não apenas problemas na execução de operações do sistema, mas também problemas de entendimento do usuário, ou seja, rupturas nesta comunicação. Por isto, o *designer* deve não apenas informar aos usuários como executar suas tarefas em condições normais, mas também como remediar potenciais problemas. Assim, os modelos de *design* de IHC devem conter elementos que permitam ao *designer* representar não apenas estas situações problemáticas, mas também os diálogos que o usuário poderá travar com o preposto para remediá-las (por exemplo, o diálogo que o usuário deve travar com o preposto para tentar corrigir os erros em um formulário que ele tenha preenchido).

Este trabalho propõe um modelo de interação que faz uso destes conceitos, visando assim apoiar o *designer* na concepção de sistemas interativos à luz da Engenharia Semiótica.

3.2 Análise Baseada em Cenários

Como dito no capítulo 1, as fases do processo de desenvolvimento onde as representações de IHC podem se encaixar são (Figura 1.1): análise (cenários e modelo de tarefas), especificação de requisitos/usabilidade (cenários e modelo de tarefas) e concepção e representação do *design* (modelo de interação). Apesar de o foco deste trabalho ser o modelo de interação, é fundamental definir quais são os elementos necessários aos cenários e à modelagem de tarefas para que estas representações possam ser utilizadas na Engenharia Semiótica, e para que sirvam de insumo para o modelo de interação proposto.

Como foi visto no capítulo 1, cenários são representados através de narrativas ricas em detalhe contextual, envolvendo usuários, processos e dados que podem ser reais ou potenciais (Figura 2.1). Apesar de contextualizados, os cenários, quando usados na fase de análise e especificação de requisitos, não devem conter detalhes da interface propriamente dita, como textos e rótulos, seleção de *widgets*[7], etc. Pretende-se com isto evitar um comprometimento precoce dos *designer*s com uma solução de interface a ser adotada, o que dificultaria a exploração de soluções alternativas que emergissem da modelagem de tarefas e do projeto cuidadoso da interação.

Através dos cenários, pode-se identificar os signos que farão parte da aplicação, desde os signos pertencentes ao domínio até os que surgiram porque as tarefas do usuário estão sendo informatizadas. A seguir será proposta uma forma de se organizar os signos que vão sendo identificados durante o processo de *design*.

Tabela de Signos

[7] *Widgets* são componentes interativos de uma interface gráfica.

As tarefas descritas nos cenários apresentam ou manipulam informações. As informações apresentadas ao usuário ou modificadas por ele são signos que aparecerão na interface de alguma forma. Pode-se classificar estes signos como signos de domínio, transformados ou de aplicação, de acordo com o grau de familiaridade que se espera que os usuários tenham com eles. Signos transportados diretamente do mundo do usuário para a aplicação são representadas por signos do domínio (por exemplo, nome e endereço). Signos originados no domínio mas que aparecem na interface através de alguma transformação, tal como analogias ou metáforas, são representados como signos transformados (por exemplo, pastas na área de trabalho do Windows). Por último, signos que só fazem sentido dentro do sistema, e não têm prévio significado para os usuários, são chamados de signos da aplicação (por exemplo, login e senha).

É importante classificar os signos em tipos (domínio, transformado ou aplicação), pois tipos diferentes requerem diferentes tomadas de decisão por parte do *designer*. Em geral, signos transportados diretamente do mundo dos usuários (de domínio) devem apenas requer informações se limitações forem impostas pela aplicação. Por exemplo, o signo *nome* é claro para os usuários, mas pode requerer a seguinte explicação – "Por favor, forneça seu nome completo. Não é permitido mais que 100 caracteres.". Signos que são transformações de signos existentes no mundo dos usuários (transformados), requerem informações sobre os limites da analogia ou metáfora realizada para transportá-los para a aplicação. Estas informações são necessárias para que os usuários entendam quais aspectos da analogia ou metáfora estão sendo considerados e quais devem ser descartados, evitando-se mal entendidos (para uma discussão sobre vantagens e desvantagens do uso de analogias e metáforas, consultar (Halasz e Moran, 1982)) Por exemplo, uma explicação sobre as pastas em um *desktop* seria "Estas pastas funcionam como no mundo real, exceto que nunca ficam cheias. Isto é, você pode manter e colocar muitas coisas nelas. Na realidade, o disco do seu computador é o real local onde as coisas estão sendo guardadas. Então, é ele que controla a quantidade de espaço que pode ser ocupado." Finalmente, signos existentes apenas na aplicação, que podem ser totalmente desconhecidos pelos usuários, requerem uma explicação completa sobre o que significam e como são utilizados. Por exemplo, o signo *zoom* em uma aplicação gráfica.

Alguns signos podem gerar dúvidas no momento de classificação em um dos tipos. Por exemplo, o signo senha pode ser classificado em aplicação, mas pode aparecer a seguinte dúvida "Não se pode considerar o signo senha nesta aplicação como uma analogia à assinatura, impressão digital ou algo que identifica uma única pessoa? Se sim, então também se pode classificá-lo como signo transformado.". Neste caso, fica a cargo do *designer* definir o tipo do signo, baseado nas características dos usuários e suas tarefas, e a quantidade de informação a ser fornecida. Em todo caso, a classificação auxilia o *designer* a refletir sobre a explicação a ser associada a cada signo. O importante da classificação não é o resultado (qual signo é de qual tipo), mas as conseqüências do tipo que foi escolhido.

A Figura 3.2 apresenta o cenário *Solicitação de Inscrição no Quadro de Avisos*, dando destaque aos signos extraídos deste cenário.

Cenário 2: Solicitação de Inscrição no Quadro de Avisos

Rita acaba de ser contratada pelo Renascer como funcionária, para trabalhar no atendimento às famílias. Dentre as instruções necessárias para o seu trabalho, Rita é avisada que há um Quadro de Avisos virtual que todos os membros da organização estão começando a utilizar. Rita explica que já conhece o Quadro, o qual havia acessado como visitante quando lhe ofereceram o emprego. Então lhe explicaram que, como visitante, ela só poderia acessar os avisos públicos, e agora deveria se inscrever para ter acesso às seções destinadas aos funcionários. Rita entra no Quadro e, partindo das instruções que encontra na página, fornece os dados necessários para o pedido de inscrição: seu nome completo, e-mail, e login e senha que deseja utilizar. Após finalizar o pedido, novas instruções são apresentadas a Rita, dizendo que agora ela precisa esperar que o responsável pelo Quadro cheque seus dados e efetue sua inscrição. Ela não entende direito como saberá quando sua inscrição será efetivada, e ao perguntar sobre isto ao coordenador da sua área, ele lhe diz que ela receberá uma mensagem através do email que forneceu.

Figura 3.2: Cenário *Solicitação de Inscrição no Quadro de Avisos,* com os signos em destaque.

Examinando todos os signos extraídos dos cenários, pode-se perceber que alguns destes signos podem ser agrupados e relacionados a conceitos ou entidades do domínio e/ou da própria aplicação. Estas entidades são representadas como signos compostos. Por exemplo, nome, e-mail, login e senha podem ser agrupados no signo composto usuário.

Os signos podem ser representados em uma tabela contendo campos como nome do signo, descrição e tipo (composto, domínio, transformado ou aplicação), como pode ser visto na Tabela 1. Esta tabela de signos pode ser completada à medida que a modelagem for sendo feita e novos signos forem surgindo.

Signo composto	Usuário	
Nome do signo	**Descrição**	**Tipo**
nome	nome do usuário	Domínio
e-mail	e-mail do usuário	Domínio
login	identificação do usuário	Aplicação
senha	senha para conferência da identificação do usuário	Aplicação
Signo Composto	Aviso	
Nome do signo	**Definição**	**Tipo**
título	título do aviso	Domínio
autor	autor do aviso	Domínio
data	data de postagem do aviso	Domínio
chamada	texto do aviso (resumido)	Domínio
texto	detalhes do texto do aviso	Domínio
quem_postou	membro que postou o aviso	Aplicação
prazo	prazo de validade da exposição do aviso	Domínio
quadro_geral	indicação se o aviso deve ser apresentado no quadro geral de avisos	Domínio

Tabela 1: Parte da tabela de signos do Quadro de Avisos.

Ao fazer esta tabela, os signos estão sendo estruturados, para que em um segundo momento eles possam compor uma ontologia da aplicação. Esta ontologia permitirá revelar e tornar explícitas as relações entre os signos, apoiando a reflexão do *designer* sobre os impactos de possíveis mudanças nos signos e nas relações entre eles.

Algumas críticas ao uso de cenários se referem à freqüência com que ficam incompletos ou ambíguos. Além disto, muitas vezes fica implícito por que determinado cenário está sendo construído, dificultando a reflexão do *designer*. Para tentar solucionar estes problemas, propõe-se complementar os cenários com perguntas que revelem os intuitos do *designer* ao elaborá-los, isto é, perguntas que identifiquem os pontos que o *designer* almeja descobrir, explorar e/ou ratificar junto aos usuários ao se construir o cenário. Além de apoiar a reflexão do *designer*, estas perguntas podem evitar que os cenários fiquem incompletos ou ambíguos, ou até mesmo revelar novos elementos nos cenários (Carroll et al., 1994).

As perguntas visam explicitar a lógica de *design* do *designer* ao construir cada cenário, através, por exemplo, do seguinte questionamento: *"Por que o* designer *elaborou a pergunta X?"*. Além de auxiliar no processo de análise, estas perguntas provêem insumo precioso para o sistema de ajuda, capturando decisões sobre o conjunto de soluções dadas pelo *designer* para os usuários que de outro modo se perdem ao longo do processo de *design* (Silveira, 2002).

O *designer* pode gerar uma lista global de perguntas que seriam referenciadas nos cenários gerados. A referência pode ser feita incluindo-se o número da pergunta entre colchetes, no trecho do cenário onde se descreve o aspecto que a pergunta pretende abordar.

Como o principal objetivo da Engenharia Semiótica é a construção da mensagem do *designer* para os usuários, os cenários devem ser acompanhados da identificação das classes de usuários que poderiam participar das situações descritas. O objetivo é permitir ao *designer* verificar quais serão os ouvintes do conjunto de mensagens corresponde a cada cenário. A Figura 3.3 apresenta novamente o cenário descrito no capítulo 2, e acrescenta os papéis de usuário e perguntas correspondentes.

Cenário 1: Consulta ao Quadro de Avisos

Ana Lúcia é colaboradora da Associação Saúde-Criança Renascer [2] há vários anos. Quando tem um tempo livre, ajuda em atividades da Organização. Como está em férias, Ana resolve usar parte de seu tempo para isto. Ela resolve consultar o Quadro de Avisos para ver se estão precisando de algo específico, ou se ela tem alguma idéia diferente. Como seu contato direto é pequeno, ela tem notícias do que está acontecendo a partir dos avisos na Internet [1]. Ana quer saber do que eles estão precisando. Em vez de navegar pelas seções do Quadro [3,4], ela resolve buscar os avisos específicos sobre isto [4]. Assim, ela pode conseguir as informações que deseja de forma mais ágil [5]. Ela informa que deseja buscar avisos sobre "doação" [6], e todos os avisos que contêm esta palavra são mostrados. Ela logo se interessa por um dos pedidos, relacionado a roupas de inverno. Então ela vê os detalhes deste aviso e o imprime [7]. Antes de sair do Quadro, ela se lembra que adorou a última feira organizada pelo Renascer. Decide então consultar a seção de Eventos para buscar informações sobre as novas feiras que serão organizadas. Para saber as últimas notícias relacionadas a este assunto, ela ordena os avisos apresentados por data, listando-os a partir do último [8]. Com isto, ela verifica que na próxima semana ocorrerá uma nova feira.

Papéis:

membro (Ana Lúcia)

Perguntas:

1.É útil permitir acesso via Internet?
2.Quem deve ter acesso ao Quadro?
3.Como são organizados os avisos?
4.De que forma posso acessá-los?
5.É importante prover um mecanismo de busca?
6.Que tipos de busca devem estar disponíveis?
7.A apresentação de um aviso deve levar em conta a possibilidade de imprimi-lo?
8.Deve haver operações para manipular a visualização de avisos? Quais?

Figura 3.3: Papéis e perguntas associadas a um cenário.

3.3 Modelo de Tarefas Adaptado

O modelo de tarefas é tipicamente construído a partir da análise do domínio do problema e do contexto do usuário. Esta análise é realizada principalmente através de questionários, entrevistas e estudos etnográficos (Preece et al., 1994 e Beyer e Holtzblatt, 1998). Com base na análise realizada, geralmente são construídos os cenários de uso contextualizados (como descrito na seção 3.2), utilizados como insumo para a modelagem das tarefas.

Para o modelo de tarefas contemplar elementos adequados à modelagem das tarefas, sem incluir detalhes da interação ou da interface, foi feito neste trabalho uma simples adaptação à s*tructure chart notation*. Esta notação representa a decomposição hierárquica de funções em seus componentes funcionais, indicando algumas relações temporais entre elas (por exemplo, funções seqüenciais, alternativas e iterativas). Na Figura 3.4 pode-se verificar a modelagem da função *Decidir se palavra está errada,* quando se ativa um corretor ortográfico, utilizando a *structure chart notation*. A adaptação feita neste trabalho limitou o *structure chart notation* a representar apenas as metas a serem atingidas pelos usuários através da aplicação que está sendo projetada e sua decomposição hierárquica em tarefas.

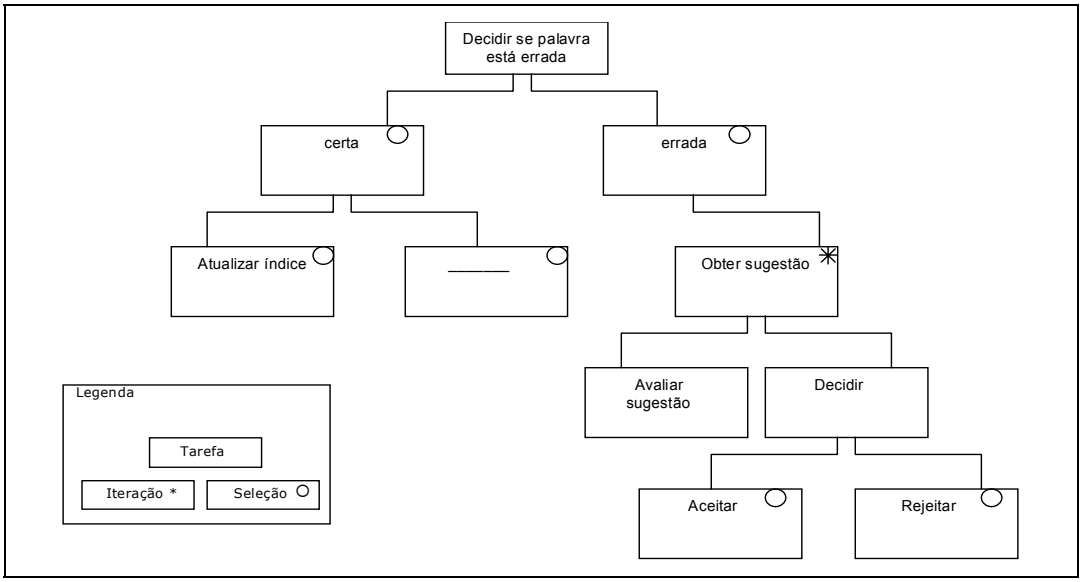

Figura 3.4: Modelagem da função *Decidir se palavra está errada* utilizando a *structure chart notation* (exemplo descrito em Preece et al., 1994).

Neste trabalho, adaptou-se a *structure chart notation* para incluir estruturas de tarefas que não seguem uma seqüência predefinida; tarefas que podem ser efetuadas em qualquer ponto da realização da meta; pré-condições para a realização de tarefas ou metas, tarefas opcionais, operadores e mecanismos para re-uso de trechos do modelo. Embora alguns destes elementos já tenham sido contemplados em outras notações existentes, optou-se por fazer uma adaptação à

structure chart notation para manter o modelo de tarefas tão simples quanto possível[8].

O primeiro passo da modelagem de tarefas consiste em extrair dos cenários as metas ou objetivos que os usuários terão ao utilizar o sistema. Cada cenário pode descrever mais de uma meta, e uma meta pode ser descrita em mais de um cenário. Sendo assim, deve-se ler todos os cenários para identificar e organizar as metas neles descritas. Para cada meta deve-se identificar os cenários que lhe deram origem, pois esta identificação facilita a rastreabilidade entre as diferentes representações utilizadas ao longo do projeto. No cenário da Figura 2.1, pode-se identificar a meta *Consultar avisos* destacada no texto por sublinhado.

Como complementação da modelagem das tarefas, este trabalho propõe que o conjunto de metas encontradas nos cenários seja organizado em um diagrama hierárquico. Este diagrama provê uma visão macro das metas que cada classe de usuários pode realizar, organizadas de acordo com algum critério que o *designer* ache relevante. Uma meta é representada por um retângulo com bordas arredondadas contendo o nome da meta, expresso do ponto de vista do usuário. Para isto, pode-se considerar como expressão da meta a lacuna indicada em uma fala do usuário tal como "Eu quero utilizar o sistema para <meta>". Cada meta é identificada por uma letra, e sua representação inclui também o(s) papel(papéis) de usuários que poderá atingi-la através do sistema.

No diagrama de metas do sistema de Quadro de Avisos do projeto ORÉ (ORÉ, 2003) (Figura 3.5), as metas foram agrupadas e identificou-se quatro tipos de situação de uso importantes: a solicitação de inscrição no quadro por visitantes (representada pela meta *Solicitar inscrição*), a consulta de avisos por visitantes ou membros (representada pela meta *Consultar os avisos*), a manipulação de avisos por membros (representada pela árvore *Manipular avisos*[9]) e a administração do quadro de avisos pelos administradores do Quadro (representada pela árvore

[8] Esta adaptação é análoga ao método de *Hierarchical Task Analysis* (HTA) (Annett e Duncan, 1967), que também se baseou na *structure chart notation* e é utilizada em alguns estudos de IHC.

[9] Metas utilizadas para agrupar outras metas não recebem numeração. Por exemplo, a meta *Manipular Aviso* (Figura 3.5), que foi utilizada para agrupar as metas *Postar aviso, Alterar um aviso e Remover aviso*, não recebeu numeração, pois ela não será decomposta em tarefas na modelagem de tarefas.

Administrar o quadro). Esta última situação não faz parte da *atividade-fim* do Quadro, mas compreende metas necessárias para o *suporte da aplicação*.

Atividade-fim e suporte da aplicação foram os critérios de classificação das metas adotados pelo *designer* do Quadro de Avisos. Para cada aplicação, as metas podem ser classificadas e organizadas de forma diferente, de acordo com critérios adotados pelo *designer* do sistema. Critérios como, por exemplo, quais são as metas a serem priorizadas no projeto da aplicação, quais metas gastam mais tempo para serem modeladas, a freqüência na qual se espera que cada meta seja atingida, quais se relacionam "fortemente" e precisam ficar "conectadas" no modelo de interação, ou até mesmo quais metas podem ser atingidas por determinadas classes de usuários.

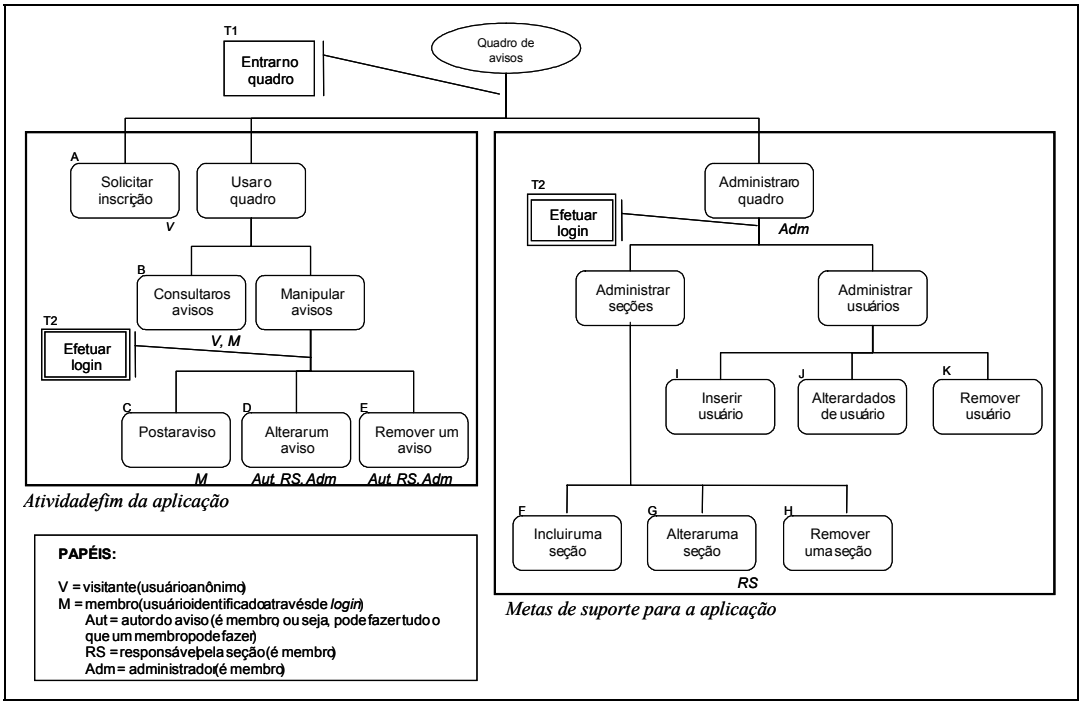

Figura 3.5: Diagrama hierárquico de metas do Quadro de Avisos.

Os papéis de uma meta (as classes de usuários que poderão realizar a meta) em um nível hierarquicamente superior no diagrama de metas são propagados para os níveis inferiores. Por exemplo, o papel "Adm" representado na meta *Administrar o quadro* será também um dos papéis de todas as metas desta sub-árvore. Caso haja papéis adicionais para uma determinada meta, estes devem ser representados na própria representação da meta, como em *Alterar uma seção*,

cujas tarefas podem ser realizadas tanto por administradores, "Adm", quanto por responsáveis pela seção, "RS".

Após a criação do diagrama, para cada meta identificada (metas no último nível - folhas - do diagrama de metas) será associado um modelo de tarefas que consiste em uma decomposição hierárquica dos passos necessários para se atingir a meta correspondente, do ponto de vista do usuário que visa atingi-la. Em outras palavras, cada tarefa pode ser decomposta em subtarefas, e cada subtarefa pode ser novamente decomposta em novas subtarefas, e assim sucessivamente (Figura 3.6). Ao representar cada meta, deve-se indicar os cenários que a descrevem (mesmo que parcialmente).

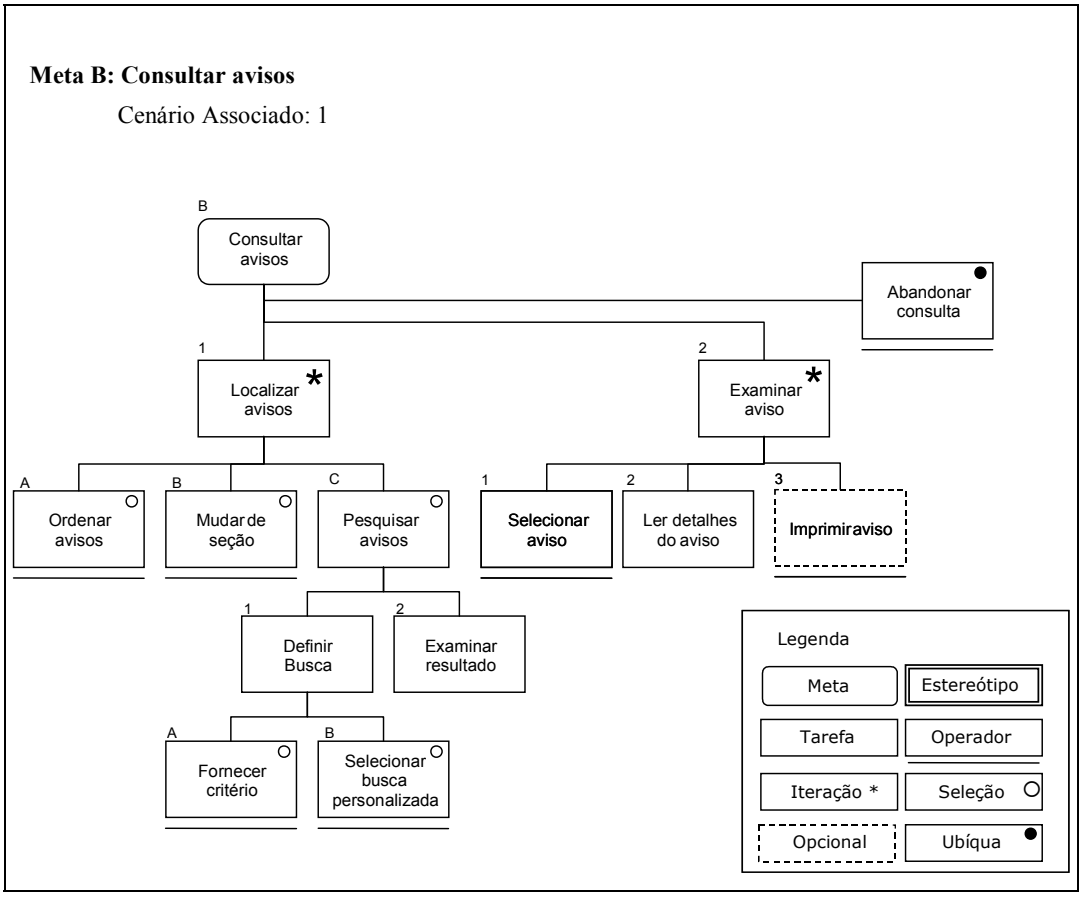

Figura 3.6: Modelo de tarefas da meta *Consultar Avisos.*

As tarefas são representadas por retângulos (por exemplo, *Localizar avisos* na Figura 3.6), com marcações especiais para indicar a que tipo de estrutura estão associadas, como será visto adiante. Deve-se ressaltar que o modelo de tarefas deve refletir como o usuário trabalha, evitando-se focar em um ambiente ou plataforma tecnológica específica. Esta consideração não apenas facilita o re-uso

de modelos de tarefas, como também evita que decisões sobre a solução de interação ou de interface sejam tomadas prematuramente, dificultando a exploração de soluções alternativas por parte dos *designers*.

Deve-se observar que as tarefas que o usuário vai realizar de fato são as que estão representadas nas folhas (último nível) da estrutura hierárquica. A decomposição das tarefas em subtarefas deve parar antes que o modelo inclua detalhes operacionais de interface, tais como "digitar X", "pressionar botão Y", etc.. Dependendo do ponto em que se pára a decomposição das tarefas, existem tarefas tão simples que poderão – em fases posteriores no processo de *design* – ser mapeadas diretamente em um elemento de interface ou de interação. Estas tarefas são representadas no modelo como operadores, os quais são representados por uma linha abaixo do retângulo. Como pode-se verificar na Figura 3.5, o *designer* optou por modelar a tarefa *Imprimir aviso* como um operador. Então, na modelagem da interação, o *designer* deve, provavelmente, estabelecer valores *default* para os signos envolvidos nesta tarefa e o usuário, ao executar esta tarefa, não precisará configurar parâmetros de impressão. Isto torna a solução mais simples e rápida para os usuários, porém menos flexível.

Estruturas de Tarefas

As tarefas podem ser organizadas nos seguintes tipos de estruturas: seqüenciais, independentes de ordem, alternativas e iterativas. Em uma estrutura seqüencial, existe uma ordem em que as tarefas devem necessariamente ser efetuadas pelo usuário. As tarefas, nesta estrutura, são representadas por retângulos contendo o nome da tarefa, expresso do ponto de vista do usuário, e um número indicando sua posição na seqüência (Figura 3.7).

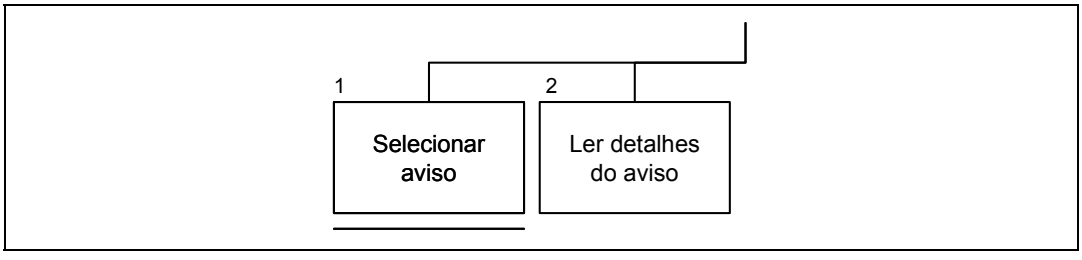

Figura 3.7: Estrutura seqüencial.

Algumas tarefas podem ser realizadas em qualquer ordem. Uma estrutura de tarefas independente de ordem representa um conjunto (e não uma seqüência) de tarefas a serem efetuadas pelo usuário. Tipicamente, o *designer* sugere uma ordem de execução, mas é o usuário quem determina, de fato, em que ordem as tarefas serão efetuadas. Neste tipo de estrutura, as tarefas são representadas como as tarefas seqüenciais mas, como a ordem é apenas sugerida, inclui-se um ponto de interrogação após o número que indica a posição relativa da tarefa na estrutura (Figura 3.8).

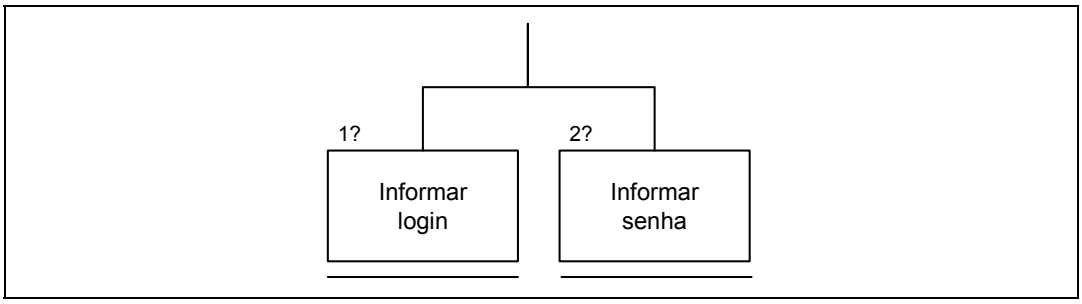

Figura 3.8: Estrutura independente de ordem.

Para se atingir uma meta, há momentos em que diversos cursos de ação são possíveis. Tais cursos de ação são representados por uma estrutura alternativa, onde o usuário deverá selecionar qual das tarefas da estrutura será efetuada. Nesta estrutura, utilizam-se pequenos círculos no canto superior direito do retângulo de cada tarefa alternativa, e letras como identificadores em vez de números (Figura 3.9).

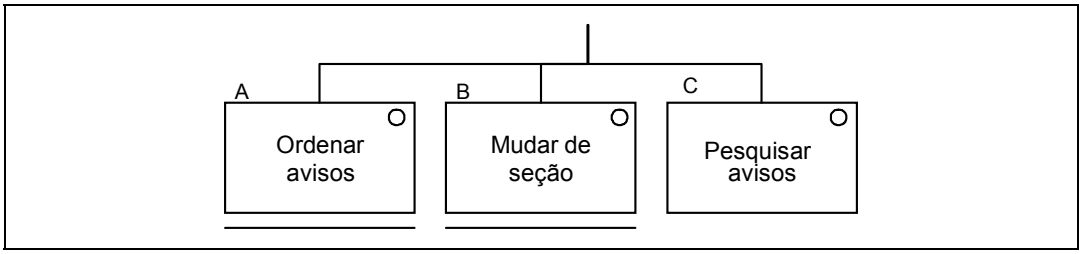

Figura 3.9: Estrutura alternativa.

Quando uma tarefa pode ser realizada diversas vezes, utiliza-se uma estrutura iterativa. Um asterisco (*) no canto superior direito do retângulo é utilizado para indicar a iteração (Figura 3.10). O número mínimo ou máximo de repetições pode ser incluído acima do retângulo e alinhado à direita. A expressão *[n+]* indica que a tarefa deve ser realizada pelo menos *n* vezes; *[m..n]* indica que a tarefa deve ser realizada no mínimo *m* e no máximo *n* vezes. Quando não houver

indicação sobre o número de repetições de uma tarefa iterativa, assume-se que seja [0+].

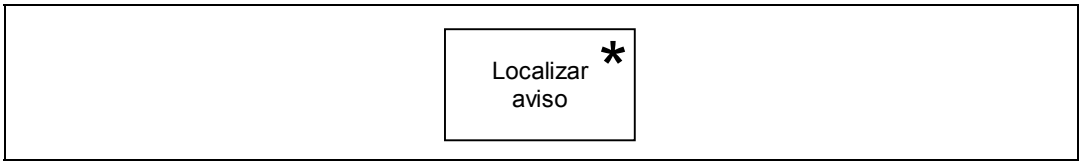

Figura 3.10: Estrutura iterativa.

A Figura 3.11 apresenta uma tarefa iterativa com suas sub-tarefas alternativas. Esta estrutura projetada pelo *designer* indica que o usuário poderá realizar a tarefa *Localizar aviso* n vezes consecutivas. Em cada vez que ele realizar esta tarefa, ele terá que optar por uma forma de localização, ou seja, através da ordenação dos avisos, da mudança da seção ou da pesquisa por avisos.

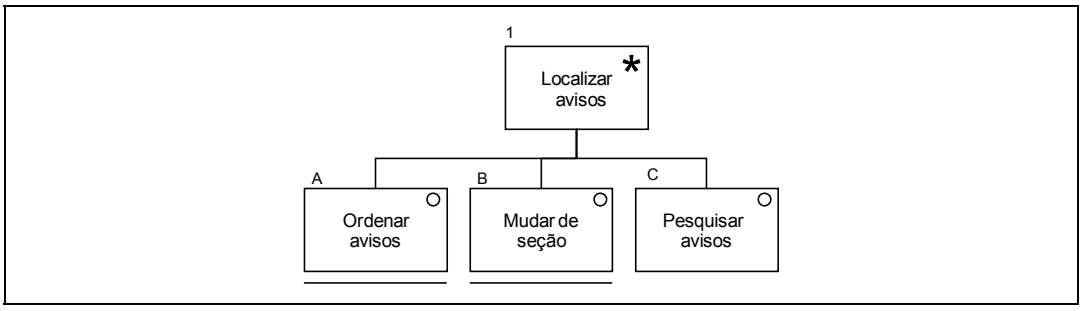

Figura 3.11: Tarefa *Localizar avisos* e suas sub-tarefas.

Tarefas Opcionais

Quando o usuário pode optar por realizar ou não uma tarefa, ela é dita opcional, e é representada com uma borda tracejada (Figura 3.12).

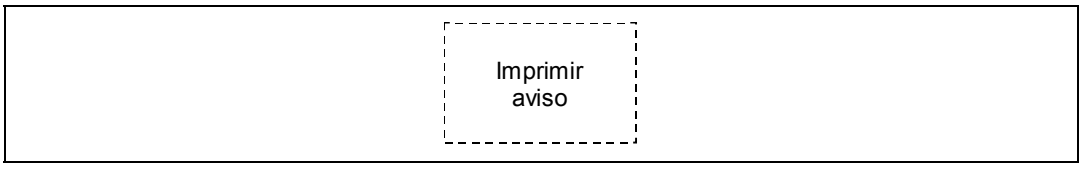

Figura 3.12: Tarefa opcional.

Tarefas Ubíquas

Algumas tarefas podem ser realizadas em qualquer ponto da realização da meta ou de determinada tarefa. Estas tarefas são denominadas ubíquas, e são representadas por um círculo preenchido no canto superior direito do retângulo (Figura 3.13). Na Figura 3.6, o *designer* modelou a tarefa *Abandonar consulta*

como ubíqua. Isto significa que o *designer* deve prover explicitamente meios para o usuário efetuar esta tarefa dentro do sistema, durante a realização da meta *Consultar avisos*, de forma independente do ambiente em que o sistema for executado (sem contar, por exemplo, com o botão "fechar janela" do Windows ou "voltar" num navegador Web).

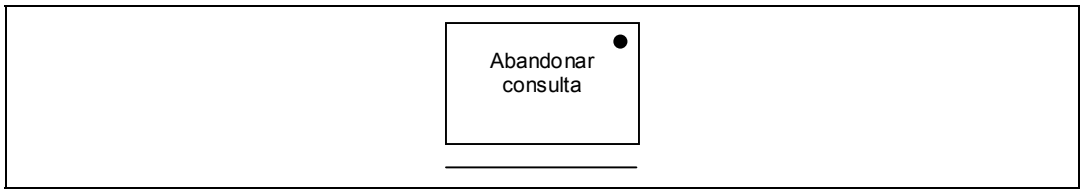

Figura 3.13: Tarefa ubíqua.

Pré-condições

É comum existirem tarefas que são pré-condições[10] para a realização de uma determinada meta ou tarefa. Estas pré-condições podem ser representadas através de um *callout* ligado a uma meta ou tarefa (Figura 3.14). No diagrama de metas da Figura 3.5, pode-se verificar que a tarefa *Efetuar login* é pré-condição para a realização de determinadas metas, como por exemplo a *Postar aviso*.

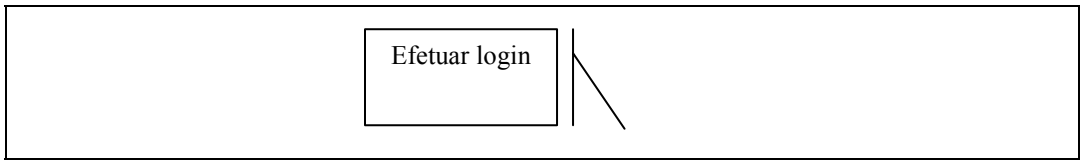

Figura 3.14: Representação de pré-condição para a realização de uma tarefa ou meta.

Re-uso de Tarefas e Metas

Há tarefas que fazem parte da estrutura de diversas metas. Neste caso, para facilitar a representação e manutenção do modelo, estas tarefas podem ser definidas como estereótipos. Um estereótipo de tarefa pode ou não receber parâmetros. Graficamente, é representado por um retângulo com borda dupla, contendo uma expressão da forma *<<nome_do_estereótipo(parâmetro1, parâmetro2, ...)>>* ou simplesmente *<<nome_do_estereótipo>>* (no caso de não haver parâmetros). A Figura 3.15 apresenta a definição do estereótipo da tarefa *Efetuar login*. Como pode-se verificar no diagrama hierárquico de metas da Figura

[10] As pré-condições que **não são tarefas**, são representadas de outra forma no modelo de tarefas e serão apresentadas mais adiante.

3.5, este estereótipo foi utilizado como pré-condição para a execução de algumas metas.

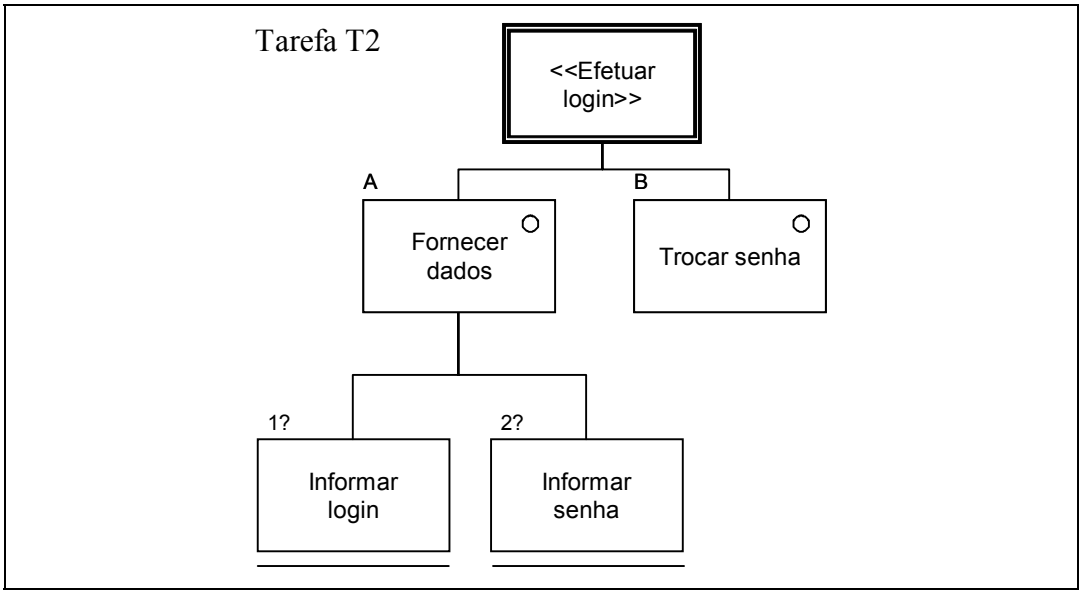

Figura 3.15: Definição do estereótipo *<<Efetuar login>>.*

Além do estereótipo, outra forma de re-uso é fazer referência direta a metas ou tarefas já utilizadas em outras partes de uma mesma modelagem, mas que não são genéricas o suficiente para se tornarem estereótipos. No caso de referência a uma meta isto é feito incluindo-se a expressão *Nome da Tarefa = META X* (onde X indica a meta em questão). No caso de referência a uma tarefa, é utilizada a expressão *Nome da Tarefa = X.Y* (onde X representa a meta e Y a tarefa em questão). E, no caso de referência a uma subtarefa, *Nome da Tarefa = X.Y.Z* (onde X representa a meta, Y a tarefa e Z a subtarefa) e assim sucessivamente, dependendo do nível onde se encontra a subtarefa. Na Figura 3.16 vê-se um exemplo de reaproveitamento da tarefa *Localizar avisos* que foi especificada na meta *Consultar avisos* (Figura 3.6).

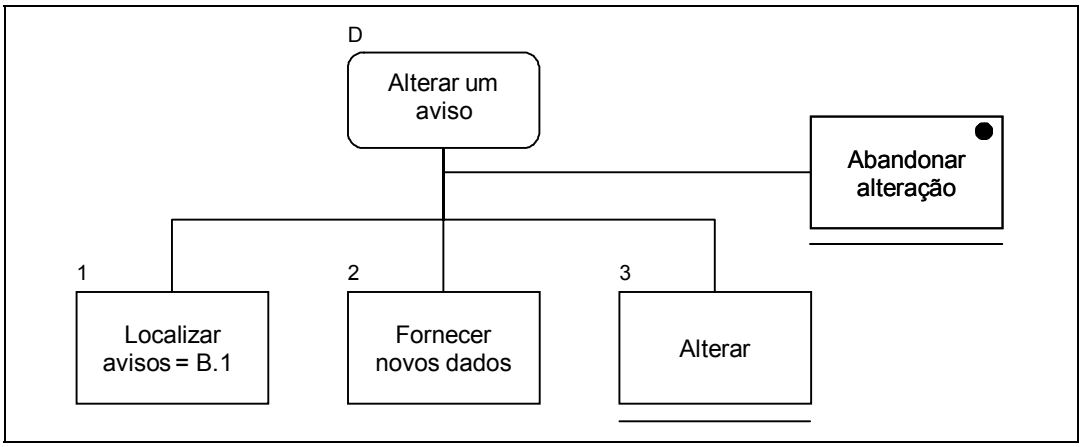

Figura 3.16: Reaproveitamento da tarefa *Localizar avisos*.

Como dito na seção 3.1, a Engenharia Semiótica ressalta a importância de se representar os signos e as rupturas na conversa preposto do *designer*-usuário que possam ocorrer durante uma interação. Por isto, além da representação diagramática de tarefas, que tem como objetivo dar uma visão geral das metas e da hierarquia de tarefas, é proposta neste trabalho uma especificação textual que inclui, para cada tarefa, os signos a ela associados e as decisões do *designer* sobre prevenção e tratamento de erros a ela vinculados.

Especificação Textual do Modelo de Tarefas

Os cenários, além de auxiliarem na construção da tabela de signos, também fornecem informações sobre quando, como e onde os signos são utilizados. Através da análise dos cenários, pode-se detectar alguns signos apresentados ou manipulados em cada tarefa. Sendo assim, após a modelagem diagramática das tarefas, associa-se às tarefas os signos correspondentes. Deve-se representar textualmente a identificação da tarefa, a palavra-chave SIGNOS, e incluir os signos correspondentes. Informações referentes ao tipo dos itens de informação (data, texto livre, etc.), seus valores *default*, ou sua obrigatoriedade serão especificadas somente no modelo de interação (capítulo 4).

Para se fazer referência a um dos signos que compõem um signo composto (vistos na seção 3.2), pode-se utilizar o formato *nome_signo_composto.nome_signo*. Se, por outro lado, for necessário fazer referência a todos os signos de uma determinada composição, pode-se utilizar o formato *nome_signo_composto.**. Por exemplo, o signo correspondente ao login

do usuário pode ser expresso como *usuário.login*, enquanto em uma situação em que são apresentados e/ou manipulados todos os dados de um usuário, o conjunto destes signos pode ser representado por *usuário.**. Caso os signos não possam ser definidos nesta fase de modelagem, ou dependam da configuração do sistema em tempo de execução, pode-se utilizar a notação *nome_signo_composto.X*.

Cada signo que for utilizado em apresentação na interface é representado na especificação textual por seu nome seguido de ponto de exclamação (*signo1!*). Já um signo que será fornecido ou manipulado pelo usuário é representado por seu nome seguido de ponto de interrogação (*signo2?*). Caso haja diversas instâncias de um mesmo tipo, pode-se utilizar o construto *conjunto(nome_do_signo,cardinalidade)* para representá-las (Figura 3.17).

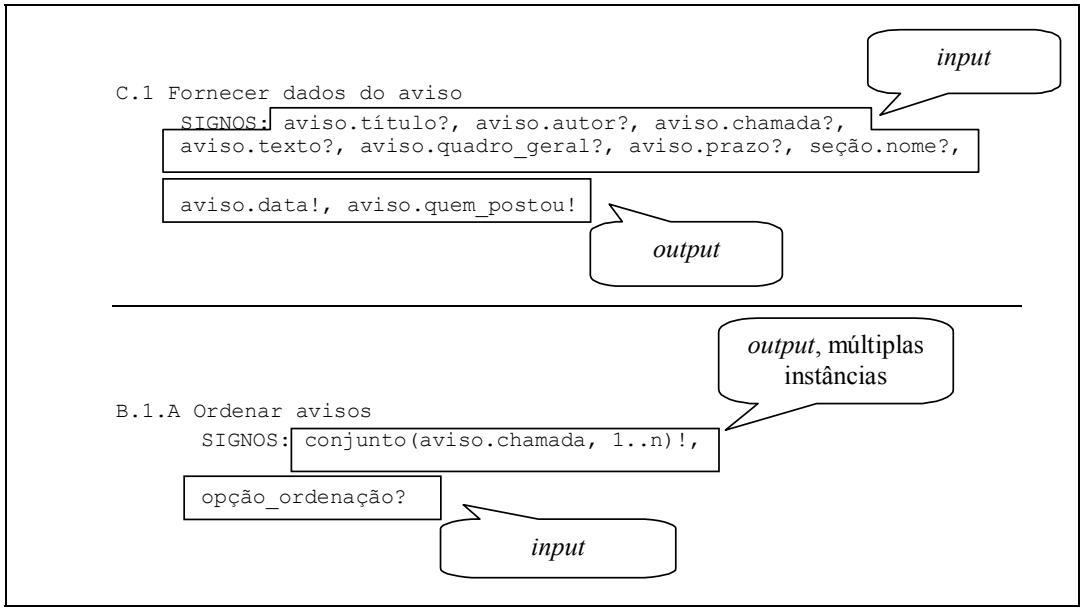

Figura 3.17: Identificação dos signos no modelo de tarefas.

Como dito anteriormente, a Engenharia Semiótica ressalta a importância de se representar os signos e rupturas que possam ocorrer durante uma interação. Por isto, além da especificação dos signos relacionados a cada tarefa, deve-se identificar as possíveis rupturas na comunicação preposto do *designer*-usuário que podem ocorrer durante uma interação, e que já podem ser previstas e registradas no modelo de tarefas. Sendo assim, para cada tarefa, deve-se especificar os tipos

de apoio a prevenção e tratamento de erro que o *designer* pretende prover, classificando-os em uma das seguintes categorias[11]:

1. prevenção passiva: Erros que devem ser prevenidos por documentação ou instruções *online* (por exemplo, "O usuário não possui acesso ao sistema").
2. prevenção ativa: Erros que devem ser prevenidos ativamente pelo sistema (por exemplo, tarefas que devem ou não devem estar disponíveis em certas situações. No modelo de interface, isto poderá ser mapeado em, por exemplo, habilitar ou desabilitar botões de acordo com o *status* atual da aplicação; impedir que o usuário digite letras ou símbolos em campos numéricos; e assim por diante).
3. prevenção apoiada: Situações que o sistema consegue detectar como sendo erros em potencial, mas cuja decisão recai sobre o usuário. Geralmente são realizados na interface por mensagens de confirmação (por exemplo, "Arquivo já existe, deseja sobrescrever?").
4. tratamento apoiado: Erros que devem ser tratados pelo usuário com apoio do preposto do *designer* (por exemplo, apresentar uma mensagem de erro e uma oportunidade para o usuário corrigi-lo).
5. captura de erro: Erros que são identificados pelo sistema e devem ser notificados ao usuário, sem que haja qualquer passo corretivo possível dentro do sistema (por exemplo, "O arquivo está corrompido." ou "Espaço em disco insuficiente.").

Os tipos de prevenção e tratamento são apresentados, no modelo de tarefas, logo após a indicação dos signos na especificação textual. Por exemplo, o TRATAMENTO APOIADO associado à tarefa *Fornecer Critério* (Figura 3.18), na especificação textual das tarefas do diagrama da Figura 3.6.

[11] Algumas destas categorias foram definidas informalmente durante o *design* de interação de um sistema para o ambiente Windows (ADDFácies, 2002).

B.1.A Ordenar avisos
SIGNOS: conjunto(aviso.chamada, 1..n)!, opção_ordenação?
B.1.B Mudar de seção
SIGNOS: seção.nome?
B.1.C.1.A Fornecer critério
SIGNOS: critério?
TRATAMENTO APOIADO: o critério deve ser informado
B.1.C.1.B Selecionar busca personalizada
SIGNOS: busca_personalizada?
TRATAMENTO APOIADO: busca deve ser informada
B.1.C.2 Examinar resultado
SIGNOS: conjunto(aviso.*, 0..n)!
B.2.1 Selecionar aviso
SIGNOS: conjunto(aviso.*, 1..n)!, aviso.*?
B.2.2 Ler detalhes do aviso
SIGNOS: aviso.*!

Figura 3.18: Especificação textual das tarefas da meta *Consultar Avisos*.

Este capítulo descreveu conceitos da Engenharia Semiótica e duas representações de *design*, cenários e modelo de tarefas, necessários ao entendimento e uso do modelo de interação proposto no próximo capítulo.