2
Trabalhos Relacionados

Neste capítulo são apresentadas representações de projetos de sistemas interativos, algumas das quais vêm sendo utilizadas para apoiar a comunicação entre os profissionais de IHC e os engenheiros de software. Estas representações serão analisadas levando-se em consideração a possibilidade de representar o conteúdo descrito na seção 1.2.1. Ou seja, a representação de:

- quem são os usuários da aplicação e o seu contexto;
- as metas que os usuários desejam alcançar através do sistema;
- visão global do conteúdo do projeto da comunicação usuário-preposto do designer necessária para o alcance das metas dos usuários, incluindo os caminhos alternativos e o tratamento de problemas na interação;
- os motivos subjacentes à solução.

2.1 Representações Típicas da Área de IHC

Na área de IHC já foram propostos vários tipos de representações para se modelar, por exemplo, o perfil dos usuários de uma aplicação, as tarefas dos usuários, a interação usuário-sistema e a interface. Estas representações podem ser utilizadas ao longo de um processo de desenvolvimento. Como o foco neste trabalho é a comunicação do projeto da interação usuário-sistema para os engenheiros de software, vamos descrever tipos de representações de IHC que contribuem ou poderiam contribuir para uma maior aproximação entre estas áreas, mas que possuem algumas limitações. São elas: cenários de uso, modelos de tarefa, modelos de interação e representações de interface. As representações de IHC concebidas de acordo com a teoria da engenharia semiótica são apresentadas ao final desta seção.

2.1.1 Cenários de Uso[7]

Cenários são geralmente descrições em forma de narrativa do que as pessoas fazem e podem/devem fazer ao utilizar o sistema interativo em construção (Carroll, 2000 e Carroll, 1995). O objetivo deste tipo de representação é facilitar a comunicação dos designers e desenvolvedores com os usuários, para que eles possam participar do processo de produção da aplicação que ele irá utilizar. A descrição dos cenários deve focar o usuário e suas tarefas (Carroll, 1995).

Para Carroll, cenários são descrições concretas, focados em instâncias particulares de uso, guiados por atividades, fragmentados, informais e abertos (isto é, possibilitam questionamentos) (Carroll, 1995). Segundo o autor, cenários podem servir de guia durante o ciclo de desenvolvimento, assumindo papéis diferentes em cada fase e com diferentes níveis de abstração. Por exemplo, na fase do projeto do software, os cenários podem ser analisados para se identificar os principais objetos do domínio necessários para se implementar estes cenários. Um exemplo de cenário para um sistema de "Base de Conhecimento" pode ser encontrado na Figura 2.

Isabela acabou de ingressar no programa de mestrado da PUC-Rio. Com o objetivo de conhecer o trabalho de seu grupo de pesquisa, ela decidiu acessar a base de conhecimento que o grupo utiliza para armazenar informações sobre suas pesquisas em andamento, trabalhos publicados e artigos relevantes de outros grupos de pesquisa. Isabela entrou no sistema e percebeu que ela não tinha um login e senha de acesso, mas notou que ela poderia logar como visitante. Ela procurou pelos documentos de sua área de interesse, e o sistema retornou uma lista de documentos armazenados. Ela examinou a lista, observando os títulos e o resumo de cada um, e optou por ler inicialmente o que parecia ser o mais simples, pois ela ainda está se habituando ao seu novo tópico de pesquisa.

Figura 2: Cenário "Busca de documentos em uma base de conhecimento."

Através dos cenários é possível descrever as características dos usuários da aplicação (no cenário da Figura 2: estudante de pós-graduação), o contexto de uso (toda a contextualização presente nos cenários indicam possíveis situações de uso da aplicação) e os caminhos para se atingir uma meta (os preferenciais, alternativos e as exceções). Além

[7] Este tipo de representação também é utilizado nas áreas de engenharia de requisitos e de software, mas devido a sua grande importância para a área de IHC, resolvemos descrevê-lo nesta seção.

disto, o designer pode explicitar no texto dos cenários os motivos das decisões que estão representadas ali. Por exemplo, na Figura 2, o designer poderia ter explicitado ali, ou mesmo em outro cenário, que o sistema fornece a opção de logar como visitante, pois os administradores da base de conhecimento são os próprios alunos e que estes demoram para verificar os novos pedidos de acesso à base e a obtenção de um login. Então, para os novos alunos não ficarem muito tempo sem o acesso a informações importantes para um novato em um grupo de pesquisa, optou-se por criar o login "visitante".

Breitman e Leite (2000) propuseram uma forma de conectar os cenários em uma rede complexa de relações, com o objetivo de gerenciar a evolução dos mesmos ao longo do processo de desenvolvimento. A conexão entre os cenários é apresentada de forma gráfica, formando-se um mapa de cenários.

Estes autores, em seu trabalho, utilizaram a estrutura para a construção de cenários proposta por Leite e co-autores (Leite et al., 1997) para a definição das relações entre eles. Segue a estrutura: título (identifica o cenário), metas (indica o objetivo a ser alcançado), contexto (descreve a localização geográfica, temporal, pré-condições e/ou o estado inicial do cenário), recursos (descreve os objetos físicos e informações que precisam estar disponíveis para o cenário), atores (uma pessoa, um sistema ou uma estrutura organizacional que tem um papel no cenário), episódio (conjunto de ações que detalham o cenário, descrevendo o seu comportamento), exceção (obstáculos no alcance da metas descritas nos cenários) e restrição no contexto ou episódio (informações sobre restrições, por exemplo, requisitos de qualidade).

As relações definidas pelos autores (Breitman e Leite, 2000) são: complemento (cenários que compartilham uma mesma meta e que apresentam alguma coincidência em seus contextos e recursos), equivalência (cenários que compartilham uma mesma meta e que apresentam alguma coincidência em seus contextos), subconjunto (cenários que compartilham o mesmo contexto, ou pelo menos, o contexto de um é completamente contido no contexto do outro e, pelo menos um episódio do cenário filho está presente no cenário pai), pré-condição (*A* é pré-condição para *B*, se *A* aparecer no contexto de *B* e existir a coincidência de pelo menos um ator entre eles), *detour* (relação definida quando um ator é mandado do cenário que possui uma exceção, para um segundo cenário que trata

esta exceção e manda o ator de volta para o primeiro cenário), exceção (um cenário possui uma exceção e esta exceção o leva para outro cenário), inclusão (conecta um cenário, que contenha um comportamento utilizado por vários outros cenários, a estes outros cenários) e possível precedência (um cenário pode ser a pré-condição para outro cenário ocorrer).

2.1.1.1 Cenários: Limitações e Pontos a serem Endereçados

Apesar de os cenários facilitarem o entendimento do problema e da solução que está sendo proposta, pela equipe de desenvolvimento e pelos *stakeholders* da aplicação, eles possuem limitações no apoio à comunicação entre os profissionais de IHC e os engenheiros de software. Como já foi dito, cada cenário representa um fragmento, um "pedaço" da aplicação. Para se representar uma aplicação inteira, são necessários muitos cenários. Essa fragmentação dificulta a visualização global de todo o projeto de IHC pelos engenheiros de software, podendo comprometer o entendimento deles sobre as decisões de IHC que estão representadas. No exemplo da Figura 2, o engenheiro de software poderia se perguntar "é possível imprimir, a partir deste ponto de interação, este documento escolhido pela usuária?". Para tentar achar a resposta a esta pergunta, ele teria que ler outros cenários. Este tipo de relação não está explícito.

Como foi visto, Breitman e Leite propuseram uma forma de se relacionar os cenários criados ao longo do processo de desenvolvimento. Entretanto, mesmo com a utilização das relações propostas por eles para se conectar os cenários, não é fácil se ter uma visão global da aplicação, pois, para isto, os engenheiros de software (ou mesmo os designers de IHC) têm que ler cada cenário, ir "navegando" através das suas relações e ir criando mentalmente o mapa da aplicação como um todo. Por exemplo, para achar a resposta para a questão da possibilidade de impressão, o engenheiro de software teria que verificar quais cenários possuem uma relação com este da Figura 2, e ler cada um até achar a resposta.

Outra limitação dos cenários no apoio à comunicação sobre o projeto de IHC para os engenheiros de software diz respeito ao seu conteúdo. Como foi dito, a intenção é que eles sejam abertos e foquem instâncias particulares de uso da aplicação. Desta forma, cada cenário foca, em detalhes, situações específicas, com o objetivo de se trabalhar (questionar)

aquele uso da aplicação. Esta visão do projeto de IHC não facilita o entendimento do que realmente deverá ser a camada de interação por parte dos engenheiros de software (Seffah et al., 2005, p. 46).

2.1.2 Modelos de Tarefa

Modelos de tarefa são representações comumente utilizadas na área de IHC, geralmente com um dos seguintes objetivos: facilitar a compreensão de um domínio de aplicação, registrar os resultados de discussões interdisciplinares, projetar novas aplicações de forma consistente com o modelo conceitual dos usuários e analisar e avaliar a usabilidade de sistemas interativos (Paternò, 2000). Alguns modelos de tarefas privilegiam a compreensão do domínio, tais como *Goals, Operators, Methods and Selection Rules* (GOMS) (Card et al., 1983), *Task Knowledge Structures* (TKS) (Johnson et al., 1988), e *Groupware Task Analysis* (GTA) (van der Veer, 1996), enquanto outros se concentram no projeto e especificação de aplicações, tais como *ConcurTaskTrees* (CTT) (Paternò, 2000) e modelo hierárquico de tarefas (Preece et al., 2002).

Geralmente, estes modelos de tarefa voltados para o projeto da aplicação utilizam representações hierárquicas, onde cada meta que o usuário pretende alcançar através do sistema é decomposta em um conjunto de tarefas que os usuários podem/devem realizar através do sistema interativo (Preece et al., 2002). Essas tarefas são relacionadas entre si, com indicações de, por exemplo, obrigatoriedade, pré-condições e papéis de usuários que podem alcançá-la.

Um exemplo simples de modelo de tarefa pode ser visto na Figura 3. O usuário tem como meta “Buscar um documento em uma base de conhecimento”. Essa meta foi decomposta pelo designer de IHC nas seguintes tarefas seqüenciais: “Informar critérios para a busca”, “Examinar resultado”, “Escolher o documento” e “Visualizar documento”.

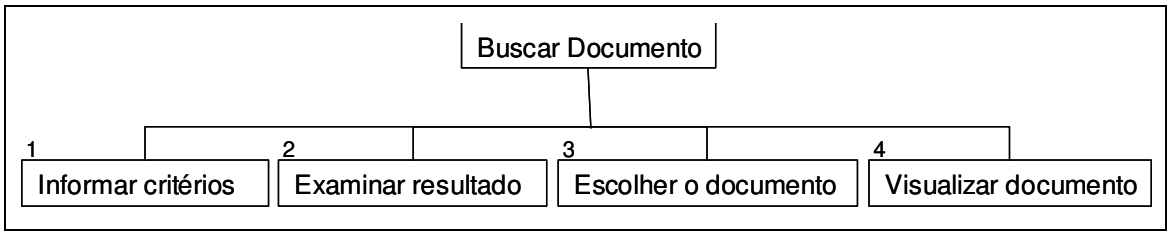

Figura 3: Modelo hierárquico de tarefas da meta "Buscar um documento em uma base de conhecimento".

2.1.2.1 Modelos de Tarefa: Limitações e Pontos a serem Endereçados

Os modelos de tarefa são úteis para se entender e explorar cada meta que o usuário pode alcançar, mas, como os cenários, são fragmentados. Para cada meta existe um modelo de tarefa. As relações entre as metas não são representadas neste tipo de modelo (e nem é esse o objetivo). Isto é, visualizando todos os modelos de tarefa de uma aplicação, não é possível explicitamente identificar a interação, ou "navegação" entre metas. Não é possível verificar se, por exemplo, o usuário estando nas tarefas da meta A, ele pode suspender esta meta temporariamente, atingir a meta B, e depois retornar para a meta A no ponto onde ele parou. Novamente, não é possível ter a noção do mapa global de interações usuário-sistema da aplicação.

Além desta limitação apontada acima, geralmente os modelos de tarefa não representam a prevenção ou o tratamento dos erros que podem ocorrer enquanto o usuário estiver realizando uma tarefa. Para cada meta, eles especificam o fluxo "desejado", ou até mesmo alternativo, para se atingi-la, mas eles não encorajam a representação de possíveis problemas na interação do usuário com o sistema durante o alcance de uma meta.

A partir de um modelo de tarefa, existem várias formas de se modelar a interação. Suponha que a modelagem da Figura 3 sirva como base para o entendimento do engenheiro de software sobre o projeto de IHC e para o seu trabalho de especificação das funcionalidades do software. O engenheiro de software não terá informações de como o usuário irá atingir esta meta via sistema interativo. Ou seja, decisões como as seguintes estarão pendentes: "O usuário poderá visualizar um documento e depois, no mesmo momento de interação com esta meta, voltar para a lista de documentos encontrados, ou

esta lista não será mantida?" ou ainda, "A lista de documentos encontrados será mantida mesmo se o usuário mude de contexto de interação, por exemplo, vá para a meta de inserção de documentos?"; "Se o usuário informar um critério de busca incorreto (erro ortográfico, por exemplo), o sistema irá ajudá-lo a se recuperar do erro, ou simplesmente irá informar que nenhum resultado foi encontrado?"; e assim por diante.

Além das questões levantadas acima, os modelos de tarefa não possuem a descrição dos perfis dos usuários, o contexto onde a aplicação será utilizada e nem a explicitação dos motivos subjacentes à solução representada. Estes conteúdos, como foi dito na seção 1.2.1, precisam ser comunicados aos engenheiros de software.

2.1.3 Modelos de Interação

Modelos de interação, como o próprio nome diz, têm o objetivo de modelar a interação do usuário com o sistema interativo. Grande parte das notações para se modelar a interação, existentes na área de IHC, possui um nível de abstração muito baixo, isto é, preocupam-se com os elementos de interface que serão utilizados na interação com a aplicação e com detalhes operacionais. Isto é, elas se concentram na especificação da interface propriamente dita (telas, *widgets*, cliques, arrastos do mouse e outros componentes de interface).

Um exemplo de modelo de interação bastante conhecido na área de IHC é a *User Action Notation* (UAN) (Hix e Hartson, 1993), que tem como objetivo representar a interação do usuário com interfaces de manipulação direta. Nesta modelagem são representadas as ações físicas do usuário na interface, os *feedbacks* da interface e/ou estado da interface após determinada ação. Alguns elementos dessa notação podem ser vistos na Tabela 2 e a modelagem da tarefa "Selecionar compromisso em uma agenda", que faz parte da meta do usuário "Alterar compromisso", pode ser visto na Tabela 3.

Elemento	Significado
~ [objeto]	Mover o cursor até o objeto
M	Mouse
Mv	Pressionar o botão do mouse
M^	Liberar o botão do mouse
objeto!	Objeto em destaque

Tabela 2: Alguns elementos da notação UAN.

Tarefa: Selecionar compromisso		
Ação do usuário	***Feedback* da Interface**	**Estado da Interface**
~[célula do compromisso] Mv	célula do compromisso!	célula selecionada
M^		

Tabela 3: Modelagem da tarefa *Selecionar compromisso* utilizando UAN (extraída de Paula, 2003).

Outro exemplo de modelo de interação é o *Instrumental Interaction* (Beaudouin-Lafon, 2000). Este modelo descreve a interação com interfaces gráficas como sendo a interação entre o usuário e objetos de domínio, mediada por instrumentos. Alguns exemplos de instrumentos são *scrollbars*, *drag & drops* e bordas de janelas. Além disto, este modelo auxilia a escolha das técnicas de interação gráfica mais adequadas para o problema em questão.

Como pôde ser visto, a modelagem da interação geralmente se concentra na especificação da interação "física" do usuário com o sistema interativo, isto é, a especificação da interação do usuário via elementos de interface concretos (mouse, *widgets*, etc).

2.1.3.1 Modelos de Interação: Limitações e Pontos a serem Endereçados

Apesar de a especificação da interação "física" do usuário com o sistema ser importante para o projeto de IHC, existe um passo anterior de especificação que não pode ser esquecido. Isto é, a especificação de todos os caminhos de interação usuário-sistema, num nível de abstração mais alto, para se ter uma visão de todas as possíveis interações que o usuário poderá ter com a aplicação. Este tipo de especificação, sem conter detalhes da interface, auxilia o designer a verificar possíveis inconsistências em sua modelagem.

Como já foi dito, o objetivo deste trabalho é a comunicação do projeto da interação para os engenheiros de software, com o intuito de que eles entendam o discurso interativo proposto pelo designer de IHC. Este tipo de modelo de interação citado acima não consegue passar de forma fácil esta informação para os engenheiros de software devido ao seu baixo nível de abstração. Eles não estão focados na definição do conteúdo da interação e sim na expressão deste conteúdo na interface, através de *widgets*, cliques, etc. Além disto, como foi apresentado na Figura 7, geralmente a especificação também se dá por meta ou tarefa do usuário, ou seja, o problema da fragmentação também ocorre com os modelos de interação. Para se ter uma visão global da interação, tem que ler a especificação de cada tarefa e criar "mentalmente" o vínculo entre elas.

Outra limitação destes modelos é que eles não incentivam a representação de possíveis problemas na interação do usuário com o sistema e de seus tratamentos. Geralmente, só é representada a forma correta, do ponto de vista do designer, de se interagir com a aplicação.

Como os modelos de tarefa, os modelos de interação também não fornecem recursos para se descrever os perfis dos usuários, o contexto onde a aplicação será utilizada e para a explicitação dos motivos subjacentes à solução representada.

2.1.4 Representações de Interface

Através das representações de interface é possível especificar a interface concreta do sistema interativo. Geralmente, para isto, utiliza-se *storyboards* e protótipos (Landay e Myers, 1996; Preece et al., 2002). S*toryboards* são desenhos das telas que representam estados da interface, já com a especificação de seus elementos (textos, gráficos, *widgets* etc). Eles podem só conter os esboços das telas individuais, mas também podem mostram a seqüência (ou parte dela) da interação entre as telas. Os protótipos também são representações de telas, mas geralmente são construídos utilizando software e já possuem algum nível de interação. Todos os dois tipos de representação são muito utilizados para se fazer a comunicação entre a equipe de projeto, e dela com os usuários da aplicação.

Então, através de s*toryboards* que apresentam a seqüência da interação e de protótipos é possível especificar os caminhos preferenciais de interação, os alternativos e os necessários para se recuperar de problemas ocorridos durante a interação.

2.1.4.1 Representações de Interface: Limitações e Pontos a serem Endereçados

Os esboços das telas individuais não deixam explícitos os possíveis caminhos de interação. O engenheiro de software (ou mesmo o designer de IHC e o usuário), para ter uma noção destes caminhos, precisa percorrer os desenhos e fazer uma simulação "mental" das possibilidades de interação.

Os s*toryboards*, quando apresentam a seqüência da interação entre as telas, não deixam explícito qual é o modelo de tarefas que está por trás deles, o seu contexto de uso, os possíveis usuários e, portanto, o que levou o designer a propor a interface desta forma. Isto também ocorre com os protótipos. Ou seja, o conteúdo da interação usuário-sistema não é o foco deste tipo de representação, e sim a expressão deste conteúdo em termos de interface abstrata ou mesmo concreta.

O engenheiro de software, ao utilizar um protótipo completo da aplicação (o melhor caso), isto é, com a representação de todos os caminhos de interação, pode, em algum momento, não perceber ou mesmo entender de forma equivocada uma decisão de, por exemplo, a criação de caminho de interação. Acontecendo isto, a interpretação do engenheiro de software sobre o discurso interativo não estará sendo compatível com a intenção do designer de IHC.

2.1.5 Representações de IHC ancoradas na Engenharia Semiótica

Como foi visto na Introdução, a teoria da engenharia semiótica vê a interação usuário-sistema como uma conversa entre o usuário e um representante do designer que está cristalizado na interface – o preposto do designer. Ao longo do tempo, várias representações foram propostas ancoradas nesta teoria (de Souza, 2005), mas quatro em

especial se concentram especificamente no projeto desta conversa usuário-preposto do designer. São elas: Linguagem de Especificação da Mensagem do Designer (LEMD) (Leite e de Souza, 1999), Linguagem de Especificação de Cenários de Interação (LECI) (Dahis, 2001), *Interactive Message Modeling Language* (IMML) (Leite, 2003 e Neto e Leite, 2006) e *Modeling Language for Interaction as Conversation* (MoLIC) (Barbosa et al., 2002[8], Barbosa e Paula, 2003, Paula, 2003 e Silva, 2005).

Nesta tese vamos nos concentrar nas duas linguagens mais recentes, a IMML e a MoLIC.

2.1.5.1 IMML

O objetivo da IMML é apoiar a especificação da interface como uma mensagem interativa, ou seja, como uma mensagem enviada pelo designer aos usuários da aplicação. Para isto, ela fornece um modelo semântico que guia o designer na elaboração da interface de uma forma abstrata e estruturada, focando o que ele quer comunicar aos usuários.

A IMML era composta inicialmente por um modelo de domínio, um modelo de interação e um modelo de apresentação (Leite, 2003). Resumidamente, o modelo de domínio descreve os objetos do domínio e as funções que mudam o estado destes objetos. O modelo de interação representa as ações realizadas pelos usuários para comandar uma ou mais funções, sendo composto basicamente pelos seguintes elementos: comandos de função (ações do usuário), resultados de função (saídas do sistema ou mensagens de erro e alertas), tarefas (composição estruturada de comandos de funções e resultados) e estruturas de interação (elementos responsáveis por organizar, por exemplo, os comandos e os resultados de função dentro de uma tarefa). O modelo de apresentação representa a mensagem global que o designer constrói para comunicar a interação para o usuário, isto é, ele organiza a apresentação dos elementos de domínio e de interação. Ele agrupa, por exemplo, em um mesmo painel de comando, algumas funções de domínio.

[8] Nesta publicação, a linguagem para se modelar a interação ainda não tinha sido denominada MoLIC.

Os modelos da IMML são representações textuais, usando elementos como, por exemplo, <*Application-Function name= "Imprimir"*> (definição do nome da função) e <*Domain-object name = "Número de cópias"* /> (definição de objetos) (Leite, 2003).

A IMML foi estendida para poder ser utilizada em processos de desenvolvimento de interfaces para múltiplas plataformas (Neto e Leite, 2006), ganhando uma nova definição no modelo de apresentação, agora chamado de modelo de comunicação.

O designer de IHC, ao utilizar os modelos de domínio e interação da IMML, estará focando uma função de domínio por vez. Ou seja, as funções são decompostas separadamente. Por exemplo, se existir a função "Imprimir", ela terá o seu modelo de domínio associado e o seu modelo de interação. O modelo de comunicação possibilita ao designer pensar na composição da interface para esta função. Desta forma, a visualização da interação fica fragmentada por função.

A Figura 4 apresenta um modelo de interação da IMML para a função "*Imprimir*".

```
<Command name = "Imprimir" domain-function = "Imprimir">
<Join>
  <View> Para imprimir um documento você deve entrar com as informações...</View>
  <Sequence>
   <Join>
    <Select>
        <Enter Domain-Object = "Nome do Arquivo">
        <Select Domain-Object = "Nome do Arquivo">
    </Select>
     <Enter Domain-Object = "Número de cópias">
   </Join>
   <Select>
    <Activate Control = "Começar" Domain-Function = "Imprimir">
    <Activate Control = "Parar" Domain-Function = "Imprimir">
    <Activate Control = "Suspender" Domain-Function = "Imprimir">
    <Activate Control = "Continuar" Domain-Function = "Imprimir">
   </Select>
  </Sequence>
</Comand>
```

Figura 4: Modelo de Interação IMML para a função "Imprimir" (extraída de Leite, 2003).

2.1.5.1.1. IMML: Limitações e Pontos a serem Endereçados

Apesar de a IMML ser ancorada na engenharia semiótica, teoria na qual esta tese também está fundamentada, ela não atende completamente ao propósito de se fazer a comunicação do projeto de IHC entre as áreas de IHC e engenharia de software. Ela possui uma notação para se modelar a interação, mas esta não facilita a visualização do mapa global de todas as conversas que podem ocorrer entre o preposto do designer e o usuário e, portanto, do discurso interativo proposto pelo designer de IHC. Isto se deve ao fato de ela representar separadamente a interação necessária para se executar cada função.

Além disto, através desta linguagem, o designer de IHC não é incentivado a explicitar o motivo de algumas decisões de design, por exemplo, o agrupamento de algumas funções em um mesmo painel de comando. E, como apontado nos outros modelos de interação apresentados, a IMML também não possui elementos explícitos para se descrever o perfil dos usuários e o contexto de uso da aplicação.

2.1.5.2 MoLIC

A MoLIC é uma ferramenta epistêmica fundamentada na teoria da engenharia semiótica, concebida para apoiar os designers de IHC no projeto das possíveis conversas que podem ocorrer entre o usuário e o preposto do designer, durante a utilização do sistema interativo (Paula, 2003 e Silva, 2005). Ferramentas epistêmicas são utilizadas para aumentar o entendimento sobre um determinado problema e suas possíveis soluções (de Souza, 2005). Desta forma, têm como objetivo aumentar o conhecimento de quem está utilizando-as, apoiando as suas tomadas de decisão, e afetando, assim, a qualidade do produto final.

A primeira edição desta linguagem foi proposta por Paula (Barbosa e Paula, 2003 e Paula, 2003), sendo composta por um diagrama de metas, um diagrama de interação, uma especificação textual associada ao diagrama de interação e uma representação dos signos. Através do diagrama de metas é possível especificar as metas que os usuários terão ao utilizar o sistema e a relação entre elas. O diagrama de interação tem o objetivo de

representar uma visão global das possíveis conversas que podem ocorrer entre o usuário e o preposto do designer para alcançarem as metas representadas no diagrama de metas. Através deste diagrama é possível definir os caminhos preferenciais, alternativos e os necessários para se recuperar de possíveis problemas da interação. A especificação textual detalha as conversas representadas no diagrama de interação. E a última representação descreve os signos que podem fazer parte da conversa usuário-preposto do designer.

Silva (Silva, 2005) revisou esta primeira edição, propondo várias extensões a esta linguagem. Ele se concentrou no diagrama de interação e na especificação textual, levantando 18 questões relacionadas às limitações e lacunas existentes na primeira edição da linguagem. Nesta segunda edição foi aprimorada a semântica dos elementos do diagrama de interação segundo a teoria da engenharia semiótica, foram fornecidos alguns novos recursos para o detalhamento da interação usuário-sistema e foi dado um passo em direção à modelagem de sistemas multiusuário (a MoLIC foi concebida inicialmente para a modelagem de sistemas mono-usuário).

Um exemplo de parte de um diagrama de interação associado à meta “Buscar documento” (definida de antemão no diagrama de metas da MoLIC) em um sistema de base de conhecimento, pode ser visto na Figura 5. Resumidamente, em cada cena (“local” onde as conversas entre usuário e preposto podem acontecer), são definidos os diálogos que podem ser travados naquele momento (indicado na parte inferior da cena). Partindo de uma cena, podem sair falas do usuário que mudam o rumo da conversa. Estas falas podem provocar um processamento do sistema ou uma mudança direta de cena. Se ocorrer um processamento do sistema, o preposto do designer deve informar ao usuário as saídas deste processamento.

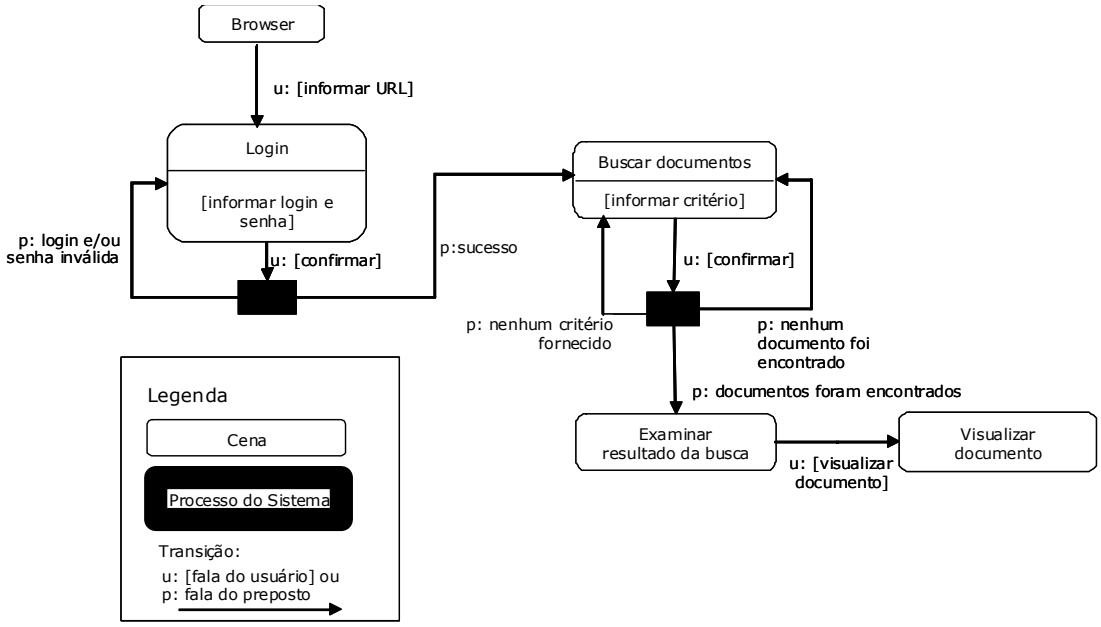

Figura 5: Parte do diagrama de interação da MoLIC relativo à meta "Buscar documento".

Associado ao diagrama de interação da MoLIC, existe uma especificação textual, que descreve as falas do usuário e do preposto que podem acontecer durante um diálogo, e as características destas falas (signo de interface associado, obrigatoriedade, possíveis valores, tipo de elemento de interface associado, etc). Vale ressaltar que os signos de interface são definidos em uma representação que também faz parte da MoLIC. Esta representação descreve as relações entre os signos, sua descrição, seu tipo (domínio, transformado ou de aplicação) e suas características inerentes.

2.1.5.2.1. MoLIC: Limitações e Pontos a serem Endereçados

Apesar de a MoLIC apresentar o mapa global de todas as conversas que podem ser travadas entre o usuário e o preposto do designer, ela possui algumas limitações para ser utilizada, da forma em que foi concebida, no apoio à comunicação entre os profissionais de IHC e os engenheiros de software. Primeiramente, ela não deixa explícito quais são os usuários da aplicação, isto é, qual é o perfil de quem irá travar as conversas que estão ali representadas. Através dela somente é possível indicar quem pode fazer o que e em que momento. Por exemplo, em um sistema de biblioteca, através da MoLIC, é possível indicar que somente o bibliotecário (e não o usuário da biblioteca) consegue travar uma conversa

com o preposto sobre o cadastramento de um livro. Entretanto, ela não fornece meios de indicar qual é o perfil do bibliotecário, ou mesmo do usuário da biblioteca.

Além dessa limitação, a MoLIC também não fornece meios para se explicitar o contexto de uso da aplicação, isto é, o ambiente onde a aplicação será utilizada. Esta informação também impacta a decisão sobre o projeto de IHC e deve ser informada aos engenheiros de software, contribuindo para o entendimento deles sobre a solução de IHC. Uma informação sobre o contexto de uso de um sistema de biblioteca seria, por exemplo, a indicação de que só existirá um terminal na biblioteca onde todos os usuários poderão consultar o acervo da mesma. Esta informação afeta fortemente o projeto de IHC, pois os caminhos de interação deverão ser pensados levando em consideração a diminuição do tempo de interação de cada usuário com o sistema interativo. Então, o engenheiro de software, ao visualizar o projeto de IHC, deve ter em mente o motivo de, por exemplo, curtos caminhos para se atingir cada meta.

2.2 Representações oriundas da Engenharia de Software

Nesta seção, primeiramente, são apresentados os *use cases* (Booch et al., 2005), representação amplamente utilizada na área de engenharia de software, e que geralmente é utilizada para se fazer a comunicação entre a equipe de projeto, e desta com os usuários da aplicação. Esta representação foi selecionada para ser apresentada aqui neste trabalho, pois na prática ela serve, muitas vezes, como insumo para o trabalho dos engenheiros de software, com o objetivo de descrever a interação do usuário com o sistema interativo.

Além dos *use cases*, é descrita a representação dos UIDs (diagramas de interação do usuário) (Vilain, 2002), que foi concebida mais especificamente na área de aplicações hipermídia. Esta representação será analisada neste trabalho, pois ela tem como objetivo representar a interação usuário-sistema necessária para se atingir as tarefas desejadas pelo usuário da aplicação.

2.2.1 *Use Cases*

Na área de engenharia de software, o paradigma mais utilizado atualmente para design, desenvolvimento e implementação de software é o de orientação a objetos (OO) (Booch et al., 2007). Baseada neste paradigma, existe a linguagem UML (*Unified Modeling Language*) (Booch et al., 2005), que tem como objetivo apoiar os engenheiros de software a especificar, construir e documentar uma aplicação. Esta linguagem possui um conjunto de modelos que permite a representação do sistema em diversas perspectivas. Um destes modelos é o *use case*, que tem como objetivo apoiar a representação dos conjuntos de ações do sistema necessários para se produzir resultados relevantes para os atores do sistema (Booch et al., 2000, p. 220). Ator é um papel que tipicamente estimula/solicita ações/eventos do sistema e recebe reações, podendo ser, por exemplo, outro sistema computacional ou um papel de usuário.

Na especificação de um sistema, existe um conjunto de *use cases*, cada qual com a representação de um ator, envolvido na interação que o modelo representa, e com a identificação das ações do sistema necessárias para o atendimento da requisição desse ator. Cada *use case* poderá ter variantes, se relacionando através de três tipos de relações com outros *use cases* (relações de generalização, inclusão e extensão). Além disto, eles são representados graficamente por elipses, contendo um nome que representam "algum comportamento encontrado no vocabulário do sistema cuja modelagem se está fazendo." (Booch et al., 2000, p. 220). O objetivo final da modelagem com *use cases* é especificar todas as funções/ações que o sistema deve prover.

Pode-se associar aos *use cases* fluxos de eventos de forma textual, indicando quando ele inicia e termina, quando ele interage com outros *use cases*, quais são os objetos transferidos e os fluxos básico, alternativo e excepcional de comportamento (Booch et al., 2000, p. 222).

Os *use cases* de um sistema podem ser conectados em um diagrama através de relações do tipo: generalização, dependência ou associação. O objetivo deste diagrama é modelar a visão estática do *use case* do sistema (Booch et al., 2000, p. 233). Um exemplo de diagrama de *use case* para um sistema de telefone celular pode ser visto na Figura 6.

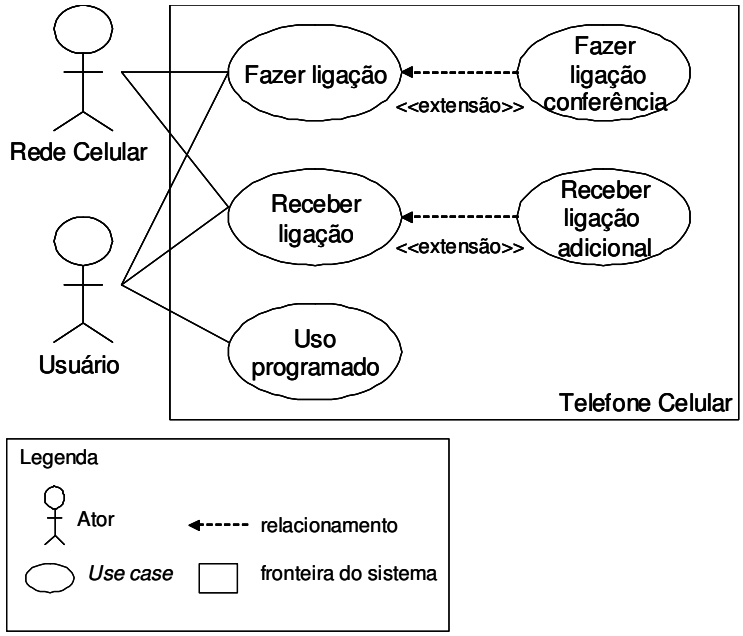

Figura 6: Diagrama de *use case* para o sistema de Telefone Celular (extraída de Booch et al., 2000, p. 232).

2.2.1.1 *Use Cases*: Limitações e Pontos a serem Endereçados

Como foi visto, o objetivo final dos *use cases* é a definição das funcionalidades que o sistema deve prover, isto é, o comportamento pretendido do sistema. Apesar de existir a possibilidade de se descrever os fluxos de eventos, não é o foco deste tipo de representação a especificação da interação do usuário com o sistema interativo, ou seja, do projeto de IHC.

Este tipo de representação não se preocupa em representar o perfil dos usuários, o contexto de uso da aplicação, a ligação explícita entre as metas dos usuários e nem os motivos subjacentes à solução que ela representa. E, apesar da possibilidade de se conectar os *use cases* da aplicação, a modelagem é fragmentada e dificulta a visão do comportamento global esperado do sistema.

2.2.2 UID

A representação dos UIDs faz parte da *Object Oriented Hipermedia Design Methodology* (OOHDM), que é um método orientado a objetos para desenvolvimento de aplicações hipermídia. Este método é composto das seguintes fases: levantamento de requisitos, modelo conceitual, projeto navegacional, projeto abstrato da interface e implementação (Rossi, 1996).

O objetivo da fase de levantamento de requisitos é obter informações sobre o domínio da aplicação. Para isto técnicas como entrevistas são utilizadas, com a finalidade de se definir os atores e as tarefas que estes pretendem executar através do sistema. Após estas definições, cenários (representando textualmente os detalhes das tarefas) e *use cases* (especificando a interação típica entre ator-sistema) são criados. Para cada *use case* especificado, um diagrama de interação do usuário (UID) é criado. Este diagrama é um gráfico que representa a interação usuário-sistema necessária para se atingir a tarefa especificada no *use case* relacionado. Ele não apresenta detalhes de interface, pois seu foco é a navegação e a troca de informações usuário-sistema em determinados momentos de interação. Um exemplo de UID para a tarefa "Seleção de um CD de acordo com o título" pode ser visto na Figura 7.

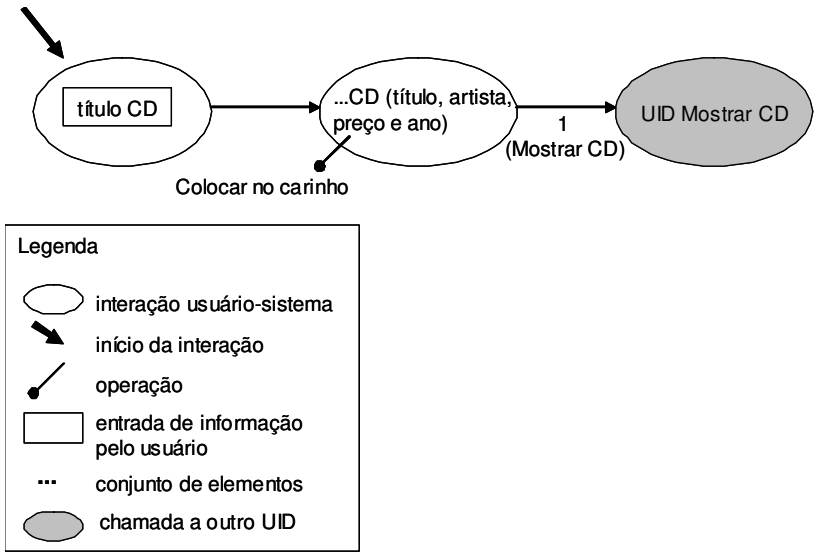

Figura 7: UID para a tarefa "Seleção de um CD de acordo com o título".

Após a criação de um UID para cada *use case*, deve-se definir a relação entre os UIDs, isto é, construir o diagrama de relacionamentos entre UIDs. Eles podem ser conectados através dos seguintes tipos de relações: inclui (um UID inclui o outro), estende (um UID pode ser estendido por outro devido a um comportamento alternativo ou opcional) e precede (um UID só pode ser executado se um outro for executado com sucesso). A Figura 8 apresenta parte de um diagrama de UIDs do sistema de "Loja de CD". O diagrama utiliza a relação de inclusão, que indica que as interações definidas no UID incluído (no caso, "Mostrar informações do cantor") não fazem sentido de serem executadas sozinhas.

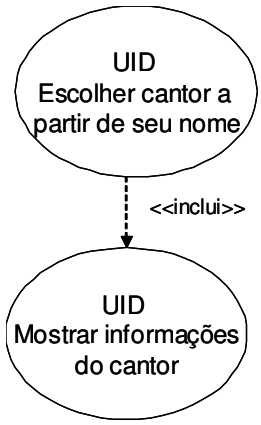

Figura 8: Exemplo de diagrama de relacionamentos entre UIDs.

Com os UIDs prontos e sua validação pelos usuários, pode-se definir as classes da aplicação, seus atributos e relações, aplicando-se regras sobre estes diagramas (Vilain et al., 2000). Tendo isto, o modelo conceitual pode ser criado, pois ele representa exatamente essas informações.

O projeto navegacional possui um conjunto de representações e tem como objetivo definir as informações que serão apresentadas aos usuários e como ocorrerá a navegação entre estas informações para que determinada tarefa possa ser executada (Güell et al., 2000). São definidas as visões navegacionais sobre o modelo conceitual. Baseando-se no modelo navegacional, cria-se o projeto abstrato da interface. Este projeto contém a definição de como serão percebidos os objetos de interface, suas propriedades e transformações. Após todas estas definições, e validações com os usuários, parte-se para a implementação da aplicação.

2.2.2.1 UID: Limitações e Pontos a serem Endereçados

Como foi visto acima, a representação da OOHDM que se propõe a fazer a especificação da interação é o UID. A separação em momentos de interação (elipses representadas na Figura 7) é definida de acordo com a ocorrência de manipulação de dados;.e as operações que o usuário pode executar são definidas sobre os itens de dados (na Figura 7, operação "Colocar no carrinho" é ligada ao item "CD"). Ou seja, o foco da modelagem são os dados, mais especificamente, os objetos que poderão ser manipulados durante a interação com o sistema.

Esta característica dos UIDs gera pelo menos duas limitações no uso desta representação no apoio à comunicação sobre o projeto da interação aos engenheiros de software. A primeira é que a definição dos momentos de interação é feita pela ótica do processamento de dados e não na ótica do usuário, ou seja, de como ele poderá perceber sua conversa com o sistema interativo. Desta forma, os engenheiros de software, ao utilizarem os UIDs, não terão acesso ao discurso interativo formulado pelo designer de IHC, e sim à especificação dos dados e as operações que poderão ser executadas sobre eles.

Além disto, apesar de existirem formas de se interligar os UIDs (inclui, estende e precede), o mapa global da interação não é facilmente visível. Cada UID está relacionado a um *use case*. Então, a questão da fragmentação também é um fator neste tipo de representação.

Através dos UIDs não é possível representar os perfis de usuário e nem o contexto de uso da aplicação. E, por último, os motivos subjacentes à solução que está representada não são explicitados.

2.3 Representações que fazem a Integração de Aspectos de IHC com a Engenharia de Software

Existem na literatura, alguns trabalhos que focam na integração de aspectos da interação humano-computador com a modelagem orientada a objetos, amplamente utilizada pelos engenheiros de software. Estes trabalhos foram analisados e agrupados em:

- extensão da linguagem UML para se incorporar a modelagem da interface (Kruchten et al., 2001; Nóbrega et al., 2005; Nunes, 2001 e 2004; Paternò, 2001; Silva, 2002);
- definição de um método que inclui modelos OO e de IHC para a especificação da interface concreta (van Harmelen, 2001);
- utilização de modelo(s) que compreende(m) aspectos de IHC para guiar a modelagem OO (Constantine, 2001; Rosson et al., 2001).

2.3.1 Extensão da UML

Silva (2002) propôs a UMLi – *Unified Modelling Language for Interactive Applications* – com o objetivo de adaptar a UML para que ela abranja a modelagem da interface (UMLi, 2007). Neste trabalho, o autor apresenta um estudo de alguns *Model-Based User Interface Development Environments* (MB-UIDEs), com o objetivo de verificar quais são os modelos de interface que usualmente são descritos através destes ambientes. Através desde estudo, o autor identificou quatro categorias de modelos: *application model* (descreve as propriedades da aplicação relevantes para a interface), *task-dialogue model* (descreve as tarefas do usuário e suas relações), *abstract presentation model* (descreve conceitualmente a estrutura visual da interface) e o *concrete presentation model* (descreve em detalhes a parte visual da interface). A partir destas quatro categorias e de um estudo de caso onde um sistema de biblioteca foi modelado com a UML, e as dificuldades para se modelar aspectos relativos à interface foram identificados, o autor estende os modelos e o metamodelo da UML, criando a UMLi.

Nunes (2001, 2004) também estende a UML para incorporar aspectos de IHC. Ele pretende através do seu trabalho colocar aspectos da engenharia de usabilidade dentro da engenharia de software. Para isto, ele propõe o método de desenvolvimento Wisdom – *Whitewater Interactive System Development with Object Models*. Este método compreende: [1] um processo de desenvolvimento centrado no usuário, evolução e prototipação rápida, [2] uma arquitetura que introduz novos modelos na UML para suportar a modelagem de

IHC (como por exemplo, modelo de interação, de diálogo e de apresentação) e [3] um conjunto de notações, que é uma extensão à UML baseada na arquitetura Wisdom.

Este mesmo autor, também possui um trabalho onde ele examina a relação da notação para modelar as tarefas do usuário, a *ConcurTaskTrees* (CTT) (Paternò, 2000), amplamente utilizada na área de IHC, com o diagrama de atividades da UML 2.0 (Booch et al., 2005). Neste trabalho, os autores propõem a incorporação da modelagem CTT à UML, utilizando mecanismos de adaptação e extensão, providos pela própria UML, no diagrama de atividades (Nóbrega et al., 2005). Segundo os autores, o objetivo é levar a modelagem de tarefas para a engenharia de software, promovendo a troca de artefatos entre IHC e engenharia de software. Vale ressaltar que o próprio autor da notação CTT (Paternò, 2000), também já propôs uma abordagem para se integrar esta notação com a UML (Paternò, 2001).

Kruchten e co-autores (Kruchten et al., 2001) propõem a extensão dos *use cases* com o objetivo de se fornecer as informações necessárias para o design da interface através da UML. Os autores propõem o uso dos *use case storyboards* - uma descrição conceitual e lógica de como um *use case* deverá ser concretizado na interface. Este *storyboard* é representado na UML através de uma colaboração <<*use case storyboard*>>. Cada *use case storyboard* contém basicamente: uma descrição textual de alto nível contendo a interação usuário-sistema relacionada ao *use case*, diagramas de seqüência e colaboração descrevendo como o *use case* será "realizado" na interface em termos da colaboração entre os objetos e atores, uma descrição de todos os requisitos de usabilidade que precisam ser levados em consideração, e explicações adicionais sobre a criação do protótipo da interface. Os autores descrevem como os *use case storyboards* podem contribuir para a construção de protótipos e da interface final (eles definem alguns passos para se construir o protótipo a partir dos elementos dos *use case storyboards*).

2.3.1.1 Extensão da UML: Limitações e Pontos a serem Endereçados

Estes trabalhos onde a solução para a integração de IHC e OO é a inclusão da modelagem de IHC dentro da UML apresentam alguns problemas. Primeiro, a UML foi

concebida para se representar aspectos internos do sistema, onde estes são decompostos em objetos e suas relações. Então, utilizá-la com um outro objetivo, isto é, modelar os aspectos de IHC, onde a decomposição destes aspectos geralmente se dá de acordo com o que será observado pelo usuário, foge do foco e propósito desta linguagem (Seffah et al., 2005, p. 47).

Segundo, os designers de IHC nem sempre estão familiarizados com o paradigma OO. Então, quando modelos de IHC passam a ser substituídos por modelos OO, isto pode atrapalhar o trabalho destes profissionais, que terão que aprender a fundo este paradigma, seus conceitos e sua forma de decomposição do problema, ou senão poderão utilizar de forma inadequada a linguagem.

E, por último, como a UML é uma linguagem utilizada por engenheiros de software, a sua extensão para incorporar aspectos de IHC pode passar a falsa impressão de que engenheiros de software estão habilitados para cuidar de todos os aspectos de IHC. Assim, trabalhos como estes, não ressaltam a importância da presença de profissionais de IHC na equipe de desenvolvimento de software.

2.3.2 Método de Design OO para a Criação da Interface

Van Harmelen (2001) apresenta o método de design Idiom, que incorpora técnicas para a especificação de interface através de *object modeling*. O autor apresenta um *framework* que descreve o uso de modelos OO e de IHC durante o design de interface. Os modelos utilizados são: cenários, modelo de tarefas, modelo de domínio (representa os objetos retirados dos cenários e as associações entre estes objetos), *core model* (representa apenas os objetos e associações que interessam ao usuário) e *view model* (provê uma visão abstrata de como o usuário interage com o sistema).

Após a construção de todos estes modelos, é feito o projeto da interface, que contém as representações concretas dos objetos do *core model.* Com o projeto da interface pronto, é construído o protótipo, e a análise e o design OO da parte interna do sistema podem ser

feitos (todos os modelos criados mais a interface concreta podem servir de insumo para o resto do desenvolvimento).

2.3.2.1 Método de Design OO para a Criação da Interface: Limitações e Pontos a serem Endereçados

Como pode ser visto, o autor propõe o uso de modelos OO para se projetar a interface. Similarmente aos trabalhos que estendem a UML, os profissionais de IHC têm que aprender OO, e sua forma de decomposição do problema, para modelarem a interface através do Idiom. Outra questão é que o autor não deixa claro qual o ganho de se modelar a interface numa perspectiva OO. Ele não descreve como estes modelos irão colaborar com ou alimentar os modelos de engenharia de software. Ele apenas menciona que isto pode acontecer, por exemplo, através da geração de *use cases* a partir do modelo de tarefas utilizado na fase de construção da interface.

2.3.3 Representação de IHC como Guia da Modelagem OO

Constantine e Lockwood (2001) propõem em seu trabalho o uso de *essential use case* para guiar o design de interface e para atravessar o *gap* entre a engenharia de software e a engenharia de usabilidade. Os *essential use cases* constituem uma descrição abstrata, generalizada e livre de tecnologia da essência do problema. Esses *use cases* não contêm detalhes de interface e nem estruturas internas do software. São baseados na intenção dos usuários, e separam visualmente essas intenções das responsabilidades do sistema. Os autores sugerem marcações nas narrativas desses *use cases* para indicar, por exemplo, objetos, classes e métodos, para facilitar a construção de modelos OO. Além disto, eles propõem a criação de um mapa que relacione os *essential use cases* do sistema, através de relações do tipo *inclusion, specialization, extension, similarity* e *equivalence*.

Rosson e Carrol (2001) descrevem como os cenários de uso podem contribuir para a construção de *object models*, fornecendo informações como: quais são os objetos, suas relações e responsabilidades. Segundo os autores, na medida que os cenários de uso do

sistema forem sendo refinados durante o design, o modelo de objetos pode ir se tornando mais completo, e o *object-oriented user interface design* construído.

2.3.3.1 Representação de IHC como Guia da Modelagem OO: Limitações e Pontos a serem Endereçados

Estes dois trabalhos apresentados acima estão em linha com o que esta pesquisa pretende fornecer para apoiar a comunicação entre os profissionais de IHC e os engenheiros de software, ou seja, o uso compartilhado de representação(ões) do projeto de IHC. O trabalho de Constantine e Lockwood pretende fazer a ponte entre as áreas através da representação da interação usuário-sistema por meio de *essential use cases*, e Rosson e Carrol através do uso de cenários de uso.

Entretanto, acreditamos, como discutido em 2.2.1.1 e 2.1.1.1 , que estes tipos de representações não são suficientes para informar aos engenheiros de software o projeto de IHC. Como já foi dito, o mapa global da interação usuário-sistema não fica explícito, dificultando o entendimento dos engenheiros de software sobre o discurso interativo que está sendo proposto pelo designer de IHC. Além disto, estas representações não fornecem meios para que o designer explicite os motivos que o levaram a propor determinada solução, por exemplo, o contexto de uso e os perfis dos usuários da aplicação.

***

É importante ressaltar aqui, que existem vários outros trabalhos que se concentram na comunicação entre as áreas de IHC e engenharia de software, mas cujo foco é a integração dos processos de desenvolvimento existentes nestas duas áreas. Podemos citar, por exemplo, UPi (*Unified Process for Interactive Systems*) (Sousa, 2005), Ferre et al. (2005), *Ripple* (Pyla et al., 2005) e *Usage-Centered Software Engineering Process* (Constantine et al., 2003).

2.4 Comparação entre as representações

Acima foram apresentados trabalhos na área de IHC, engenharia de software e que têm como objetivo explícito fazer a integração de aspectos de IHC com a modelagem orientada a objetos. Como foi discutido, acreditamos que este último tipo de trabalho, isto é, que foca modelagens OO, não são suficientes para compartilhar o discurso interativo entre os designers de IHC e os engenheiros de software. Eles são muito mais voltados para a decomposição OO, e forçam que a modelagem da interação e da interface seja baseada neste paradigma. Além disto, não é objetivo deste trabalho apoiar apenas a comunicação entre profissionais de IHC e aqueles engenheiros de software que estejam utilizando este paradigma.

Nesta seção vamos focar os trabalhos relacionados a representações de IHC e engenharia de software que, apesar das limitações, podem ser considerados ao propormos uma nova ferramenta para apoiar a comunicação entre os profissionais de IHC e os engenheiros de software. A Tabela 4 apresenta um resumo do que foi descrito sobre essas representações. Nesta tabela, o atendimento de uma representação a um critério foi baseado nas notações mais conhecidas na sua respectiva área, e se elas permitem (ou não), de alguma forma, o tipo de representação descrito no critério.

Representação /Critério	**Descrição do Perfil usuários**	**Descrição do Contexto de uso**	**Descrição de Metas**	**Foco no Conteúdo do Projeto de IHC**	**Motivos Subjacentes à Solução**	**Visão Global[9]**	**Caminhos Alternativos**	**Tratamento de Problemas Interação**
Cenário	Sim	Sim	Sim	Sim	Sim	Não	Sim	Sim
Modelo de Tarefas	Não	Não	Sim	Sim	Não	Não	Sim	Não
Modelo de Interação	Não	Não	Sim	Não	Não	Não	Sim	Não
Rep. de Interface	Não	Não	Não	Não	Não	Sim	Sim	Sim
IMML	Não	Não	Sim	Sim	Não	Não	Sim	Sim
MoLIC	Não	Não	Sim	Sim	Não	Sim	Sim	Sim
Use cases	Não	Não	Sim	Não	Não	Não	Sim	Sim
UID	Não	Não	Sim	Não	Não	Não	Sim	Não

Tabela 4: Comparação das representações quanto ao atendimento dos critérios descritos na seção 1.2.1.

[9] Visão global significa poder visualizar todas as possibilidades de interação do usuário com a aplicação de uma forma não fragmentada.

Analisando a Tabela 4, podemos perceber que nenhuma das representações apresentadas atende completamente, da forma com está, a função de apoiar a comunicação do discurso interativo proposto pelo designer de IHC (como definido na seção 1.2.1) para os engenheiros de software. Desta forma, vamos priorizar os critérios e analisar se alguma das representações atende os principais critérios.

O principal objetivo da ferramenta a ser proposta é apoiar a comunicação de todo o conteúdo do projeto de IHC para os engenheiros de software (incluindo a definição dos caminhos alternativos e tratamento de problemas), como foi visto na seção 1.2. Desta forma, analisando a Tabela 4, só as seguintes representações atendem ao mesmo tempo os critérios de "foco no conteúdo do projeto de IHC", "caminhos alternativos" e "tratamento de problemas de interação": cenários, IMML e MoLIC. É interessante notar que as três representações possuem base lingüística, ou seja, cenários são narrativas textuais em linguagem natural e IMML e MoLIC são fundamentadas na teoria da engenharia semiótica, teoria que explica a interação como um fenômeno comunicativo.

Como também foi dito na seção 1.2.1, a intenção é fornecer a visão completa (não fragmentada) de todas as conversas que podem ser travadas pelo usuário ao utilizar o sistema interativo. Dentre estas três representações, a única que atende o critério "Visão Global" é a MoLIC.

Portanto, neste trabalho vamos utilizar a MoLIC como parte da ferramenta de apoio a comunicação do projeto de IHC. Os conteúdos que a MoLIC não representa, apontados na seção 2.1.5.2.1, serão considerados no momento da proposição da nova ferramenta. Ou seja, eles serão cobertos por outras partes da mesma.