2
Trabalhos Relacionados

Este capítulo apresenta uma revisão do referencial teórico envolvido na concepção deste trabalho, como as abordagens mais utilizadas para modelar interação (cenários e *storyboards*) e linguagens para o mesmo fim. Por fim, é apresentada a Engenharia Semiótica, teoria na qual a MoLIC se fundamenta.

2.1
Cenários

Um cenário é uma descrição textual de uma situação contendo **atores**, **eventos** e sequências de **ações**. Geralmente contém informações sobre o **contexto** dos atores e suposições sobre seu ambiente, suas metas e objetivos [Carroll, 1995]. Cenários são compartilhados por vários *stakeholders*[1] no projeto de uma solução e podem ser expressos como *storyboards*, demonstrações de vídeos ou simplesmente narrativas textuais.

Um dos principais objetivos de se construir cenários é, junto aos usuários, corrigir ou confirmar o entendimento dos designers sobre as tarefas a serem apoiadas pelo sistema, bem como explorar decisões alternativas de projeto. Além disso, um dos grandes benefícios na utilização de cenários é que, uma vez que eles são descrições informais, são também mais baratos de modificar e podem ser facilmente desenvolvidos, compartilhados e manipulados. Mesmo quando utilizados em conjunto com esboços, modificá-los ainda tem um custo muito menor do que fazer o mesmo em um protótipo.

Porém, justamente porque são informais, os cenários são frequentemente **ambíguos**. As especificações formais fornecem uma solução para essa ambiguidade, mas elas são difíceis de entender [Seffah et al., 2005], dificultando o seu uso por parte dos *stakeholders*.

Nos trabalhos anteriores com a MoLIC, os autores exploraram a afinidade da linguagem com cenários, já que eles podem ser vistos como mais uma forma

[1] *Stakeholders* são todas as "partes interessadas" envolvidas na execução de um projeto, incluindo clientes e usuários.

de representar a conversa usuário-sistema. Segundo Paula (2003), essa utilização pode se dar: antes da concepção e da modelagem da interação, através de cenários de análise; durante a concepção da interação, na elaboração de situações de interação alternativas; durante a modelagem, na contextualização de situações representadas na MoLIC; e após a modelagem (ou após cada etapa de modelagem), em situações a serem avaliadas pelos usuários.

2.2 Storyboarding

Storyboards, em IHC, são construídos através de uma série de desenhos ou esboços mostrando como o usuário irá interagir com o artefato sendo desenvolvido [Sharp et al., 2007]. Eles são comumente combinados com cenários, de modo a tanto descrever a interação imaginada quanto explorar contextos de uso.

Muitos projetos se baseiam na criação de *storyboards* a partir de esboços e permitem a geração de protótipos para a execução de testes. Podem ser citados o DENIM [Newman et al., 2003], que permite o esboço de páginas da *Web*, por meio de desenho à mão livre, e a inclusão de imagens, texto e *links* entre as páginas, gerando um protótipo para o teste da navegação; DEMAIS [Bailey e Konstan, 2003], que permite a construção de interfaces e a organização de *storyboards* com relações temporais e condicionais entre as telas (sob forma de anotações ao redor dos esboços) gerando protótipos e permitindo o teste da solução; e a ferramenta DAMASK [Lin e Landay, 2008], que segue o mesmo princípio do DENIM, porém apoiada com um modelo de atividades, de forma que permite ao designer tanto uma visão abstrata da interação fornecida por um modelo, quanto uma visão concreta pelas telas desenhadas.

No entanto, apesar de *storyboards* serem relativamente fáceis de acompanhar e entender, eles normalmente descrevem um mundo **idealizado**: quando tudo ocorre como planejado e nada leva o usuário a cometer erros ou seguir diferentes caminhos de interação para atingir o mesmo objetivo, por exemplo. Isso pode ser observado mesmo quando *storyboards* são apoiados por modelos que também descrevem situações "ideais", tal como acontece no DAMASK.

Algumas desvantagens foram apontadas no uso de *storyboards*, tal como relatado no estudo conduzido por Haesen em conjunto com outros autores em [Haesen et al., 2008] [Haesen et al., 2009], que revelou uma carência de notações e ferramentas para técnicas de *User-Centered Design* (UCD). Técnicas como *storyboards* e cenários foram criadas para serem propositalmente

mente informais e imprecisas, provavelmente para minimizar ou esconder detalhes que poderiam levar à perda de foco na solução interativa, por parte dos *stakeholders*.

Portanto, a falta de notações adequadas levam os designers a enfrentar problemas na tradução dos *storyboards* e cenários em notações apropriadas para desenvolvedores de software. Uma saída para abordar esse problema é encontrar e mapear *boundary objects* [Star e Griesemer, 1989], como artefatos que podem ser compartilhados por membros dos eventuais times de design e desenvolvimento: de designers da interação e interface a engenheiros de software e programadores. O próximo capítulo apresenta a linguagem MoLIC, que acreditamos ser um *boundary object* em potencial [Paula e Barbosa, 2007].

2.3
Engenharia Semiótica

Semiótica é a disciplina que estuda os processos de significação e comunicação [Eco, 1976]. Significação, por sua vez, é o processo pelo qual certos sistemas de signos[2] são estabelecidos em virtude das convenções sociais e culturais adotadas pelos usuários destes signos. Comunicação é o processo pelo qual, para uma variedade de propósitos, produtores de signos expressam significados pretendidos, explorando os sistemas de significação existentes, ou ocasionalmente, utilizando signos não sistematizados.

Ao se projetar um software, é necessária a elaboração de sistemas de significação que permitam ao usuário compreender de que formas interagir com esse software para alcançar seus objetivos. A Engenharia Semiótica é uma teoria de IHC que enxerga os artefatos computacionais como mensagens unidirecionais dos designers para os usuários, representando a solução do designer para os problemas e necessidades dos usuários, segundo a sua visão [de Souza, 2005]. A teoria classifica os signos em três tipos: estáticos (como rótulos e menus), dinâmicos (como barras de rolagem) e metalinguísticos (signos que falam sobre o próprio software como conteúdos de ajuda e tooltips).

A teoria é diferente das outras tradicionalmente adotadas em IHC, usualmente inspiradas na teoria cognitiva e focadas no aprendizado e raciocínio. A Engenharia Semiótica enquadra a interação como um processo de comunicação, antes do que de aprendizado, colocando o designer no centro do processo. No entanto, a articulação da teoria não é feita de modo a substituir

[2]Um signo é "qualquer coisa que representa algo para alguém". Peirce [Peirce, 1931] compreende um signo como uma estrutura composta por três partes: um objeto, uma representação e uma interpretação.

as demais, mas sim de complementá-las, dando ao designer um papel de destaque, informando-o sobre o quê de fato ele está comunicando aos usuários [de Souza e Leitao, 2009].

Portanto, a teoria enxerga o sistema como um artefato de metacomunicação, pois contém em si próprio a mensagem do designer para o usuário sobre como atingir seus objetivos através do software projetado por ele. Como o designer não está presente no momento da interação, os signos estáticos, dinâmicos e metalinguísticos são os únicos meios de que ele dispõe para passar essa mensagem. Essa mensagem é, por consequência, chamada de metamensagem (mensagem sobre a mensagem), e pode ser sumarizada como:

> "Esta é a minha interpretação sobre quem você é, o que eu entendi que você quer ou precisa fazer, de que formas prefere fazê-lo e por quê. Este é o sistema que eu projetei para você, e esta é a forma que você pode ou deve usá-lo para conseguir atingir os objetivos incorporados na minha visão."[de Souza, 2005] (p.84)

Para que a interface "fale pelo designer", os designers precisam passar a sua mensagem direta ou indiretamente através de um agente comunicativo no sistema, chamado de *preposto do designer*. Assim, a Engenharia Semiótica explicitamente caracteriza IHC como um processo comunicativo bi-direcional, contendo a mensagem do designer para o usuário e a interação do usuário com essa mensagem. Portanto, para a teoria, o sucesso ou fracasso de uma interface gráfica está associado a uma comunicação bem ou mal-sucedida de decisões arbitrárias de design, e não de uma escolha específica de signos de interface (*widgets*, imagens, palavras, etc).

Segundo de Souza [de Souza, 2005], a maior contribuição teórica que a Engenharia Semiótica pretende dar para IHC é endereçar necessidades tanto dos designers quanto dos usuários quando estão se comunicando através de tecnologias baseadas em computador. Para isso, a teoria propõe a utilização de ferramentas epistêmicas, que ao invés de procurar resolver o problema diretamente, visam aumentar o conhecimento de quem está lidando com o problema, munindo-o de informações sobre o problema em si e sobre suas implicações.

A Engenharia Semiótica tem diferentes tipos de ferramentas epistêmicas que apoiam as atividades centradas no conhecimento: modelos interpretativos, princípios analíticos e métodos analíticos. A teoria enfatiza os modelos e outras representações de conhecimento produzidos e utilizados em IHC, e este trabalho procura se aprofundar justamente nesse tema.

A MoLIC, criada como uma ferramenta epistêmica e apoiada na Engenharia Semiótica, tem como objetivo descrever a conversa que ocorre entre designer e usuário, apoiando a reflexão do designer sobre as mensagens que serão trocadas entre o sistema, seu preposto, e o usuário. Este trabalho procura instrumentar o designer, possibilitando que ele enriqueça a modelagem do diálogo feita na MoLIC com esboços de tela, visando melhorar a sua descrição sobre **como** o sistema veiculará sua mensagem.

2.4 Projeto de Interação Baseado em Modelos

Na literatura encontramos diversas propostas de arquiteturas baseadas em modelos feitas com o intuito de apoiar o desenvolvimento de software como um todo. Especificamente para a modelagem de interfaces e interação com o usuário, encontramos arquiteturas como o CAMELEON [Calvary et al., 2003], MECANO [Puerta, 1996] e WISDOM [Nunes e Cunha, 2001].

Esses trabalhos foram propostos principalmente visando oferecer uma experiência de uso consistente em diferentes dispositivos e mídias, ao mesmo tempo reduzindo o esforço necessário para criar interfaces específicas para cada ambiente. Essa característica é predominante no framework de referência CAMELEON, do qual originou-se a linguagem UsiXML [Limbourg et al., 2004], sendo ambos considerados em estudos conduzidos pelo *W3C Model-based User Interfaces Incubator Group*[3], grupo criado com o intuito de encaminhar a pesquisa em design de interfaces orientado a modelos para a autoria de aplicações na *Web*.

No CAMELEON e no UsiXML existe a representação de um modelo para a **interface abstrata** (AUI) – que é independente de plataforma ou dispositivo; um modelo de **interface concreta** (CUI) – dependente de dispositivo/plataforma, porém independente de linguagem; e um modelo que deriva a interface concreta em seu estado implementado (FUI).

A figura 2.1 exemplifica uma tarefa de *Download* para um determinado domínio. A primeira camada se comunica com a camada de interface abstrata onde há um controle. Este controle é instanciado de diferentes formas na camada de interface concreta – podendo ser de software (2D ou 3D) ou um controle físico (como um botão em uma máquina). A interface final é responsável por representar detalhes de cada plataforma (HTML, Desktop, VRML, etc.).

[3] `http://www.w3.org/2005/Incubator/model-based-ui`

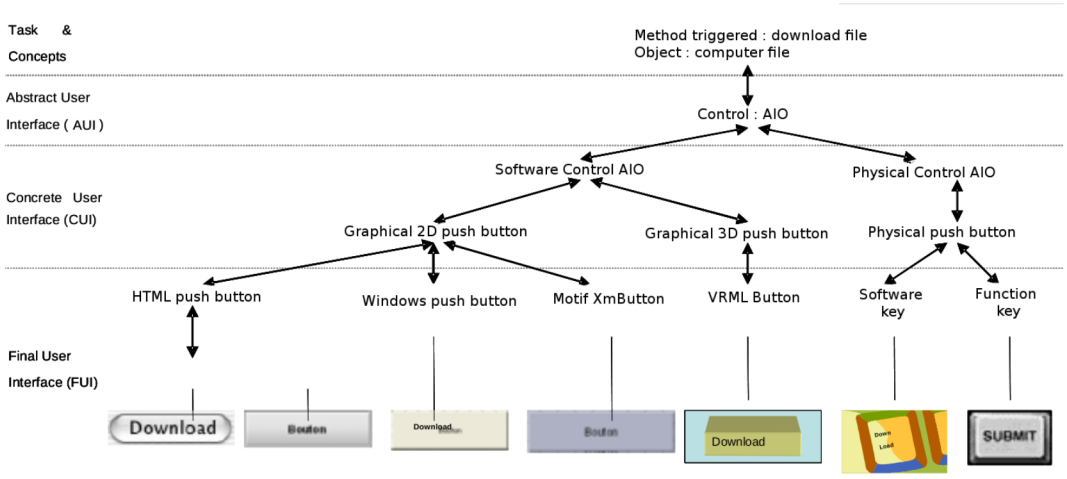

Figura 2.1: Conceitos do framework CAMELEON utilizados no UsiXML (fonte [Limbourg et al., 2004]).

Desta forma, ainda que o enfoque deste trabalho não seja o de propor ou analisar as transformações entre modelos, e sim apoiar a **atividade** de design, ainda é possível traçar uma relação do CAMELEON com a proposta de extensão da linguagem MoLIC, posicionando-a entre os modelos representados no framework.

O conceito mais próximo da representação do diálogo entre usuário e sistema apresentado no CAMELEON é o modelo de tarefas (*Tasks*), porém não há um mapeamento direto entre o este e o diálogo representado na MoLIC – que pode ser considerada como um modelo intermediário entre o modelo de tarefas e o de interface abstrata.

No CAMELEON, a linguagem utilizada para representar tarefas é o ConcurTaskTrees (CTT) [Paternò, 2000], e se concentra no projeto e especificação de aplicações, segundo um roteiro hierárquico.

A figura 2.2 exibe um modelo CTT ao lado de um diagrama MoLIC para uma tarefa hipotética de edição de um documento. Nota-se que, no CTT, o foco se dá principalmente nas relações de precedência entre as tarefas, enquanto que na MoLIC o foco é dado na troca de turnos entre usuário (u) e sistema (d) (a linguagem MoLIC é melhor explicada no próximo capítulo).

O CTT especificamente é adotado pela maioria das abordagens baseadas em modelos para IHC atualmente [Nunes, 2001]. No entanto, existem algumas lacunas na notação e em sua interpretação para as quais um trabalho com a MoLIC poderá contribuir significativamente. O tratamento de erros é uma

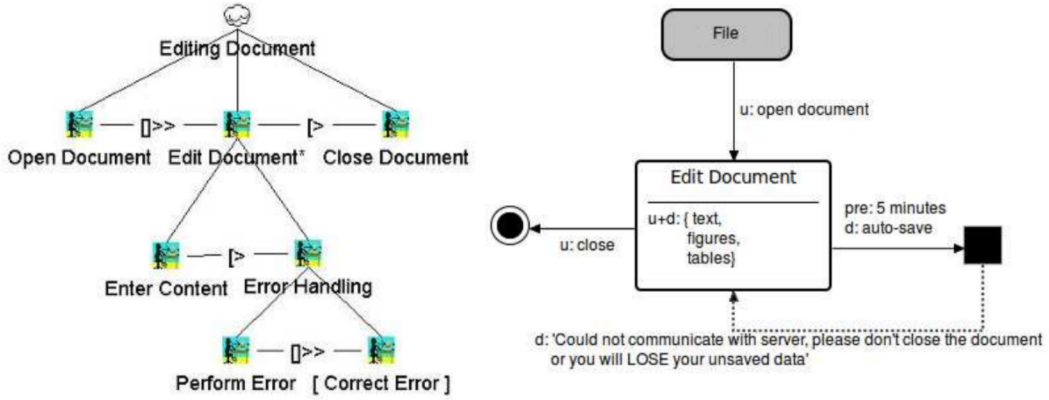

Figura 2.2: Diagrama do CTT ao lado de um diagrama MoLIC para a mesma tarefa.

delas, pois não fica claro na modelagem de tarefas, exceto talvez pelo nome dado à tarefa, quais delas servem ao propósito de recuperação de erros.

Investigando a abordagem feita pelo CTT, observamos que outra lacuna deixada por ele é a incapacidade de visualizar todos os caminhos necessários (ou ainda os preferenciais, por parte do designer) para o atingimento de todos os objetivos.

Acreditamos, portanto, que um modelo de diálogo como a MoLIC possa complementar a modelagem de tarefas, contribuindo para um grau de expressão maior, no tocante à interação da forma como ela será efetivamente experimentada pelos usuários.