5
Discussão

Neste capítulo discutimos os resultados e contribuições de nossa pesquisa. Na primeira seção apresentamos nossa interpretação sobre o que os resultados dos estudos empíricos sugerem sobre o modelo proposto. Na seção seguinte, discutimos sobre questões teóricas e conceituais referentes a EUD, Engenharia Semiótica e Sistemas de Grupo no contexto desta pesquisa.

5.1
Reflexões sobre os Estudos Empíricos

5.1.1
Golfo de Avaliação

Iniciamos os estudos empíricos desta pesquisa em 2008 quando analisamos como dois grupos distintos de usuários finais potencialmente envolvidos em situação de EUD reagiram ao uso da linguagem de macros do CoScripter para representar extensões e customizações de sistemas Web. Os resultados mostraram a viabilidade de utilizarmos essa linguagem para expressão dos signos dinâmicos da proposta de evolução do sistema. Porém, vimos a necessidade de incrementá-la, pois não representa ações relacionadas ao Golfo de Avaliação (Norman, D.A., 1986).

Podemos dizer que nosso modelo supera esta deficiência e permite aos usuários participantes da discussão expressar elementos do Golfo de Avaliação. Isto pode ser feito de duas maneiras: (i) Através da associação de "página(s sucessivas) de resposta" a uma ação de interface, mediante associação de *scripts* a um elemento de interface; (ii) ou, através da representação "textual" do que se passaria no Golfo de Avaliação, mediante comentários sobre elementos de interface ou através de descrição geral.

O caso (ii) foi evidenciado no Estudo Empírico II. Por exemplo, Paulo escreveu na descrição geral da mensagem (ver em anexo a mensagem de Paulo - Problema 2) o seguinte:

> *... Incluir no início da página a mensagem "FAVOR CONFIRMAR O ENCAMINHAMENTO"*

Outro exemplo é evidenciado na mensagem de Rute sobre o problema 2 (ver Figura 5.1). Rute inclui uma "mensagem de alerta" (Golfo de Avaliação) e ainda insere uma anotação na interface para complementar sua ideia.

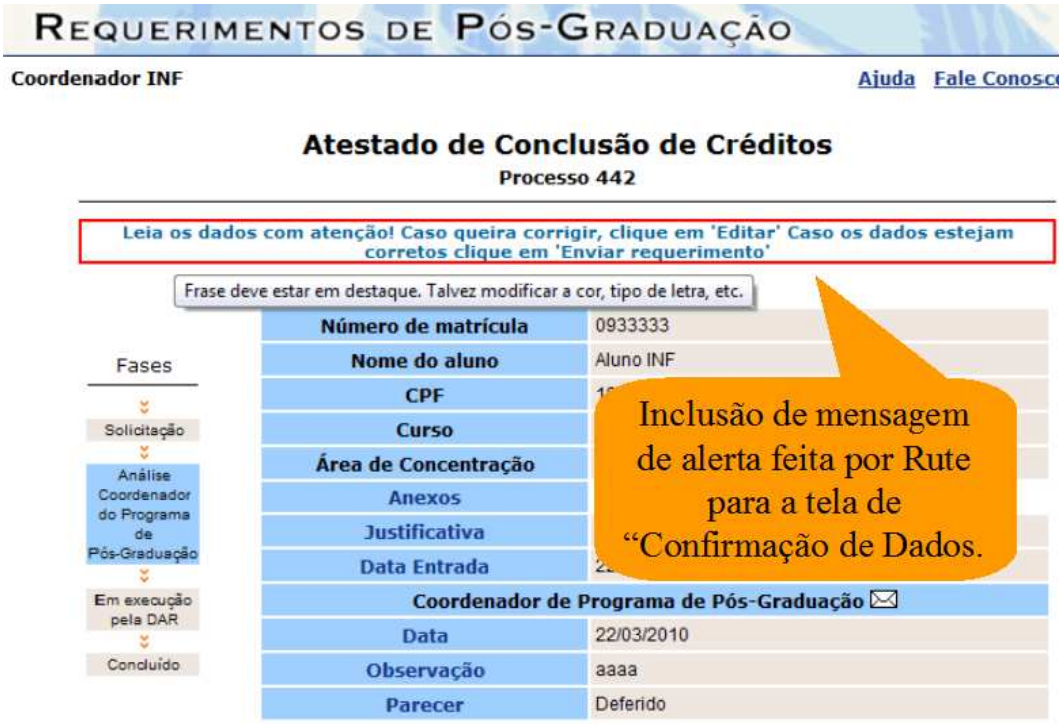

Figura 5.1: Exemplo de representação do que se passaria no Golfo de Avaliação - Recorte da página alterada por Rute como proposta de melhoria do sistema.

5.1.2 Dificuldade de Projetar, Design Reflexivo e Design Rationale

O resultado mais saliente do Estudo Empírico I, a respeito do uso da primoTiWIM foi que os participantes têm dificuldades para imaginar modificações sobre o sistema. A tarefa de "projetar" foi bastante difícil para eles. Quando tentaram melhorar o sistema, a maioria deles não tinha certeza se deveria mudar um objeto existente da interface, adicionar um novo objeto, emitir mensagens de erro ou dar mais instruções sobre a página. Eles tinham pouca noção de como as mudanças poderiam afetar usuários com outros papéis.

Esse é um dado interessante para programação feita por usuários finais, porque evidencia a necessidade de apoiar uma atividade reflexiva potencialmente longa e anterior à implementação, como idealmente deve acontecer na programação realizada por programadores de sistemas computacionais. Será que as ferramentas de EUP estão dando suporte a essa reflexão? Acreditamos que a troca de ideias com outros usuários é uma forma de reflexão e pode ajudar a amadurecer a proposta.

A Engenharia Semiótica tem um caráter epistêmico que auxilia designers a refletirem sobre a tecnologia que eles estão produzindo. No caso desta pesquisa, estamos explorando amplamente esta característica da Engenharia Semiótica porque o fato das pessoas conversarem sobre a tecnologia que eles estão produzindo, é em si, um ato epistêmico. Por um lado, a troca de ideias produz aprendizado, e por outro, o fato dos participantes terem que comunicar a ideia, já traz embutido em si um ato reflexivo.

Outra caraterística do modelo que leva a uma atividade reflexiva é o fato da comunicação ser assíncrona. Um benefício importante deste tipo de comunicação é o seu potencial para apoiar a co-construção de conhecimento através do discurso (Gilbert & Dabbagh, 2005). Gunawardena, Lowe e Anderson (1997) descrevem comunicação assíncrona como uma importante ferramenta pedagógica que "permite que grupos separados no tempo e no espaço se engajem na produção de conhecimento compartilhado." (Gunawardena et al., 1997)

O trabalho de (Sengers, 2005) fala da importância da "reflexão" em atividades de *design* de IHC. Eles argumentam que os *designers* devem refletir quais valores, atitudes e maneiras de ver o mundo estão embutidos inconscientemente na tecnologia que eles produzem, e ainda qual o efeito disto. Para endereçar esta questão os autores propuseram o "Reflective Design", um conjunto de princípios e estratégias de design que leva *designers* a repensar metáforas e valores inconscientes, embutidos na tecnologia. Além disso, a ideia do "Reflective Design" é engajar os usuários nesta mesma reflexão crítica. Os autores se referem à reflexão crítica, como o ato de trazer aspectos inconscientes de experiência para a percepção consciente. Este reflexão crítica é crucial para a liberdade individual e qualidade de vida em sociedade, visto que sem ela adotamos sem pensar atitudes, práticas, valores e identidades que não adotaríamos conscientemente. Os autores acreditam que a reflexão crítica realizada por *designers* e usuários em projetos de software é um componente essencial para um *design* tecnológico socialmente responsável.

Nosso trabalho não conduz especificamente a esse tipo de reflexão proposta em (Sengers, 2005), porém acreditamos que ele pode promover o "Reflective Design" uma vez que estamos oferecendo uma ferramenta para discutir sobre a tecnologia que está sendo construída e melhorando a comunicação entre usuários e *designers*.

Reforçando a ideia de reflexão coletiva em atividades de *design* de software, estudos anteriores de projeto de software analisaram atividades colaborativas que ocorrem em encontros presenciais, como por exemplo atividades de *brainstorm* e revisão técnica (D'Astous et al., 2001; 2004), e

identificaram vários tipos de atividades de design colaborativas.

Uma parte destas atividades está relacionada com o "objeto" de design. Elas guiam em torno da evolução do problema de *design* e sua solução; por exemplo, atividade de melhoria ou identificação de soluções alternativas e atividades de avaliação, como soluções de avaliação. Outro tipo de atividade gira em torno da construção de referências comuns, ou base comum, por um grupo de co-designers. Por exemplo, clarificação ou atividades de sincronização cognitiva ocorrem quando um grupo negocia ou constrói uma representação compartilhada do estado atual da solução. Acreditamos que podemos contribuir para este tipo de atividade, visto que nosso modelo promove a discussão partindo de uma base comum que é a interface do sistema, e permite uma construção colaborativa por um grupo de *co-designers*.

Um terceiro tipo de atividade de *design* são as atividades de gerenciamento de grupo, as quais são frequentemente relacionadas com questões de processo. Por exemplo, atividades de gerenciamento de projeto que giram em torno da coordenação de pessoas e recursos, como alocação e planejamento de tarefas. Outro exemplo deste tipo de atividade é o gerenciamento de reuniões, a ordenação e adiamento de tópicos de discussão. Este tipo de atividade não está sendo considerada em nosso modelo, o que não impede de estendê-lo neste sentido.

Barcellini et al. (Barcellini et al., 2005-A) argumentam que outra estratégia de *design* colaborativo é promover um "debate" de argumentações, onde *co-designers* realizam atividades de *design* e avaliação fazendo perguntas entre eles. Os argumentos têm uma forma muito específica e podem ser caracterizados como uma sequência de "bate e volta" ("move" and "turn"). Por exemplo, D'Astous et al. (2004) analisou os movimentos argumentativos em reuniões de revisão técnica de software. Eles descobriram, por exemplo, que a elaboração de uma solução pode ser seguida imediatamente de sua avaliação apenas, ou de sua avaliação seguida de desenvolvimento de uma solução alternativa. Estas atividades de revisão podem, ou não, serem precedidas por uma troca de sincronização cognitiva.

As argumentações tornam explícito o *design rationale*; ou seja, o raciocínio por trás do *design* de um artefato. Ao tornar seus raciocínios explícitos, *designers* têm um meio de rastrear as decisões passadas e comunicar estes raciocínios para outras pessoas fora da equipe de projetistas (Moran & Carroll, 1996).

Diferentes metodologias propuseram rastrear os *design rationales* em reuniões de design. Infelizmente, designers geralmente vêm estas metodologias como imposição de trabalho extra para eles; trabalho que não fornece benefícios

imediatos para eles. Barcellini et al. (2005-A) sugere como evitar esta objeção construindo ferramentas capazes de extrair automaticamente informações relacionadas ao *design*, partindo de arquivos de discussões online.

Discutir evolução de sistemas por meio computacional é um forma de registrar o *design rationale*. Nosso modelo registra o *rationale* da discussão em diversos níveis de abstração, desde o registro da troca de mensagens (quem falou com quem sobre o que), até o histórico da construção da representação concreta. Todas as modificações realizadas na interface são registradas. Porém, precisamos trabalhar uma forma de apresentá-las de maneira que o usuário final compreenda e também de maneira que elas possam ser referenciadas durante a discussão.

5.1.3 Metacomunicação do Modelo

Os estudos empíricos indicaram que a metacomunicação do ambiente em que a primoTiWIM opera (ie. integrado com a ferramenta de email) ainda não está tão boa quanto deveria. Uma ruptura na comunicação de João e Maria apresentada no Capítulo 4 apontou um problema de referência entre elementos da mensagem (uma parte deveria fazer referência a outra parte). A mensagem de João envolveu alteração de duas páginas, a página do coordenador e a do aluno, porém este fato não foi comunicado para Maria, que por isto não entendeu o motivo pelo qual a página do aluno havia sido anexada à mensagem:

> *"Eu nunca tinha visto essa tela na minha vida, porque a tela que eu fiz o teste era outra, então eu fiquei completamente perdida". (Fala de Maria)*

Esta fala, salienta que o ambiente de implementação parcial prejudicou uma avaliação justa dos mecanismos de referência local e global, bem como das relações retóricas que ligam falas a outras falas em um processo mais complexo de comunicação para fins de negociação. Mostra também que os marcadores de referenciação (elementos de uma comunicação que "apontam para" outros elementos ou outros trechos da própria história da comunicação) têm um papel crucial na tecnologia proposta e que o modelo proposto tem de ser estendido para oferecer mecanismos sofisticados de referenciação entre signos integrantes do processo de comunicação.

Outra evidência do problema de mecanismos de referenciação foi a desconexão dos nomes dos arquivos HTML com o que eles representam. O nome dos arquivos funciona como um "apontador" (semioticamente, um índice) que deve permitir a quem o utiliza estabelecer as conexões

lógicas entre múltiplos elementos apontados. Por exemplo: Maria propôs duas soluções alternativas para o mesmo problema, logo ela anexou duas páginas na mensagem, cada uma com uma alternativa diferente, os nomes dos arquivos são:"SugestãoAProblem1" e "SugestãoBProblema1". Esses nomes são apontadores representantes de uma relação de modalidade - cada instância sendo uma "possibilidade" alternativa para a mesma coisa. Ficou claro que precisamos dar apoio aos usuários do modelo para permitir fazer conexões lógicas eficientes entre os múltiplos objetos apontados.

Além disso, outros elementos do modelo funcionaram como apontadores. A descrição detalhada foi criada para ser um elo de referenciação entre o conteúdo geral da mensagem e o conteúdo detalhado. No estudo empírico na maioria das vezes ela foi utilizada para dizer algo como: "*Segue em anexo um arquivo .zip com o HTML da pagina alterada*" (descrição de Raquel). Outras vezes ela continha referência da quantidade de páginas envolvidas na modificação, por exemplo: "*seguem as duas telas com modificações.*" (descrição de Rute). Neste caso, Rute sinalizou que a modificação envolvia duas páginas do sistema.

A descrição geral também foi usada para fazer referência à outra parte da mensagem. Um exemplo pode ser visto na mensagem de Leo: "*[...] Sugiro a mudança do nome do botão Confirmar para pré-visualizar na tela de edição.[...]"*, só nesta frase Leo faz referência ao elemento da interface "Confirmar" e a uma página do sistema que ele chamou de "tela de edição".

As anotações também funcionaram como apontadores. Por exemplo, Leo escreveu "*Os botões Voltar e Confirmar devem ficar no centro.*". Essa anotação de Leo faz referência a dois objetos da interface: os botões Voltar e Confirmar.

Cunha (2001) evidenciou o uso intenso de apontadores (setas e símbolos de "*") nas extensões produzidas no estudo empírico, os quais partiam de elementos da interface e chegavam em textos em linguagem natural descritivos da extensão desejada. Em (Cunha, 2001) podem ser vistas várias ilustrações disto.

Identificamos um problema em nosso modelo que acontece quando existem múltiplas referências a um mesmo objeto, como por exemplo, quando um objeto da interface sofre mais de uma alteração. Uma ruptura na comunicação entre João e Maria aconteceu porque Maria não percebeu que um mesmo elemento da interface tinha sido alterado e anotado por João. Além disso, o modelo deveria deixar saber se a referência permite que se rastreie "todas as referências" feitas ao elemento durante a discussão, na ordem temporal de ocorrência, ou se é algo mais primário (só a referência corrente). Esta é uma questão a ser investigada.

O desenvolvimento de software livre é um caso particular de *design* colaborativo e assíncrono (Sack, 2006). Como analisado anteriormente por Sack et al. (2006), o projeto de software livre é realizado em três espaços de atividade: espaço de discussão, espaço de documentação e espaço de implementação, sendo que uma grande parte do processo é feita no espaço de discussão. Por esse motivo, pesquisas ((Barcellini et al., 2005-B), (Barcellini et al., 2005-A), (Mockus et al., 2000)) têm sido realizadas com o intuito de conhecer mais sobre discussões online no contexto de desenvolvimento de "software livre". Em (Barcellini et al., 2005-B) os autores estudaram a dinâmica da troca de mensagens e revelaram que é possível a construção de um discurso coletivo através de "citações" entre mensagens. A ideia é que os desenvolvedores utilizem "citações" como um mecanismo de manter o contexto da discussão. Os autores mostraram ainda que construir o discurso baseado em "citações" é mais eficiente do que a técnica convencional de "responder para", onde construção do discurso é feita através de links sucessivos de respostas de mensagens anteriores.

As "citações" dessas mensagens funcionam como marcadores de referenciação, pois ele são elementos de uma comunicação que "apontam para" outros elementos ou outros trechos do histórico da comunicação. Elas fazem a conexão lógica entre partes das mensagens. Portanto, estas pesquisas reforçam a importância de expandir nosso modelo de maneira que ele permita fazer referência entre partes da própria mensagem e a trechos do histórico da comunicação. O "tipo de fala" é um elemento do modelo que se propõe a funcionar como um conector entre mensagens, porém ele sozinho não é suficiente, existe a necessidade de outros conectores.

Gilbert e Dabbagh (2005) estudaram sobre a estrutura de discussões online em geral, não só no contexto de software livre, objetivando entender como protocolos de discussão online influenciam o discurso de uma comunicação assíncrona. O resultado revelou que alguns itens de protocolo influenciam positivamente a discussão enquanto outros podem ter impacto negativo. Existem três elementos que impactam de maneira significativa o conteúdo de uma discussão online estruturada, são eles: (a) Recomendações, (b) rubricas de avaliação e (c) itens de protocolo. Recomendações (*guidelines*) facilitadoras são regras para incentivar a participação dos interlocutores. Por exemplo, uma das regras aplicadas na pesquisa foi que as postagens deveriam ser feitas em dupla. Outra regra dizia que os participantes da discussão deveriam levantar pelo menos duas questões por semana. Rubricas de avaliação são medidas para classificar a contribuição dos participantes, como por exemplo, a frequência de acesso. Itens de protocolo são detalhes

formais como o número de postagens exigidas, o tamanho da postagem, etc. (a) e (b) influenciaram no aumento do número de postagens promovendo uma grande interação entre os participantes. Em (c), os itens "limite do tamanho da mensagem" e "requerer citação de leituras" causaram um impacto negativo sobre o discurso.

A discussão em nosso modelo também é estruturada, porém não impusemos nenhuma regra muito restritiva para elaboração da mensagem. Uma recomendação seria determinar a ordem com que os elementos do modelo devem ser preenchidos. Isto porque os estudos empíricos mostraram que o conteúdo da "descrição geral" variou de acordo com a ordem de preenchimento. Nesta pesquisa não chegamos a avaliar a dinâmica da discussão, porém já dá para perceber que é algo complexo e necessário de ser investigado.

5.1.4 Circuitos de Conversa

No Estudo empírico II apresentamos evidências de que a ordem com que o emissor descreve os dados da mensagem pode influenciar no que vai ser dito na descrição na geral. Por exemplo, se o emissor iniciar a proposta escrevendo primeiramente a "Descrição Geral", este campo tende a conter a intenção geral do que está sendo proposto. Mas se a "Descrição Geral" é escrita após a descrição detalhada na primoTiWIM, a "Descrição Geral" pode funcionar como um comentário do que foi proposto na descrição detalhada. Ou seja, na segunda situação, a narrativa é um pouco mais detalhada que na primeira. Logo, nos parece que diferentes circuitos de conversa podem estar associados a diferentes estratégias de negociação, levando a diferentes resultados.

5.1.5 Diferenças de Culturais

Um ponto que nos chamou atenção nesta pesquisa foi que a divergência de culturas (técnica e não-técnica) pode causar ruídos na comunicação entre designers e usuários. No Estudo Empírico II observamos o uso de um vocabulário específico de designers e programadores de sistemas computacionais. Por exemplo, Vitor e Raquel, respectivamente, escreveram:

> *"Apesar de não ter sido representado, o* **radio button** *indeferido deverá ficar abaixo do* **radio button** *Aguardando".(Fala de Vitor)*
> *"Sugiro mudar o* **radio button** *para um* **checkbox***". (Fala de Raquel)*

Ou seja, as palavras **radio button** e **checkbox** são palavras que fazem parte da cultura de programadores. Já no experimento com usuários finais

(Estudo Empírico I), esse fato foi pouco evidenciado. Isto aponta para a possibilidade de que podemos ter problemas na troca de mensagens entre designers e usuários finais, causado pela divergência cultural entre eles.

Susan Gasson (Gasson, 1999) identificou a necessidade de uma linguagem de design específica para a comunicação entre *designers* e pessoas que não são técnicas mas participam do processo de desenvolvimento de softwares. Seu trabalho foi realizar uma análise crítica sobre abordagens de *design* centrado no usuário (User-centered Design[1]). Neste contexto é recomendado envolver usuários do sistema no processo de desenvolvimento de software. Em seus estudos, um dos problemas percebido foi que os avaliadores de requisitos (pessoas que não são técnicas de informática mas conhecem o domínio da aplicação e por isso, participam do desenvolvimento do sistema) têm uma cultura onde o conhecimento é formalmente registrado, ou seja, a informação não existe no domínio público a menos que ela tenha circulado na forma escrita. Já os desenvolvedores técnicos evitam "documentação" escrita, eles participam de uma cultura onde o conhecimento é comunicado e validado informalmente. Gasson evidenciou que, por causa dessa divergência cultural, desenvolvedores técnicos foram capazes de ignorar o trabalho dos avaliadores de requisitos ao se recusarem a reconhecer as representações de *design* escritas pelos avaliadores. Por esse motivo, a efetiva participação dos avaliadores não aconteceu. A autora conclui que é crítica a forma na qual o conhecimento é representado, visto que expectativas sócio-culturais pré-existentes devem ser gerenciadas. Uma preocupação para ser tratada no *design* centrado no usuário é a falta de uma linguagem comum ou representação para colaboração entre grupos de diferentes áreas (disciplinas).

Em nosso modelo estamos colocando em contato os produtores e consumidores de tecnologia e estamos oferecendo uma forma de representação "concreta" da tecnologia. Portanto, podemos dizer que nosso modelo tem o potencial de endereçar esta questão de UCD. O quanto ele pode contribuir é algo a ser investigado.

5.1.6 Comparação com Outras Tecnologias

No Estudo Empírico II, Rute e Raquel refletiram sobre a diferença entre usar a primoTiWIM para comunicar a ideia ou imprimir a tela e rabiscá-

[1] "UCD is a design philosophy and a process in which the needs, wants, and limitations of end users of a product are given extensive attention at each stage of the design process." (User-Centered Design, 2010) Tradução: UCD é um processo de design no qual as necessidade, desejos e limitações dos usuários ganham extrema atenção em todas as etapas do processo de desenvolvimento de sistemas.

la. Essa questão nos dá a oportunidade de dizer que esta pesquisa se insere no contexto de sistemas de grupo, onde existem diversas pessoas envolvidas, localizadas em tempo e espaço diferentes, e com papéis distintos. Neste contexto, nem sempre é possível "dar um print na tela e ir lá no lado mostrar para pessoa". Então, principalmente neste contexto, é importante que cada interlocutor tenha um espaço para refletir e elaborar sua proposta (eg. usar a primoTiWIM), antes de mostrar aos outros. Sem falar em outros benefícios adicionais que a tecnologia proporciona, como o registro de todas as falas, por exemplo. Além disso, "dar um print na tela" não considera os papeis do sistema envolvidos na modificação.

Vitor levantou a possibilidade de usar o Balsamic (Balsamic, 2010) para expressar a ideia, ao invés da primoTiWIM. Adotar a sugestão de Vitor seria permitir a expansão ilimitada do sistema de significação do software em expansão, e consequentemente não preservaríamos a identidade da aplicação.

5.1.7 O "Poder" da Anotação

A anotação foi utilizada para registrar e comunicar a lógica de modificação, ou seja, a intenção e razão de mudança foi bem representada através deste recurso da ferramenta. Por exemplo, Pedro descreveu e justificou a remoção o botão "Desmarcar" (ver Figura 5.2).

Ela também foi usada para representar signos dinâmicos (processos). Por exemplo, Pedro escreveu a seguinte anotação no botão "Confirmar": *"Deveria ter a ideia de "passo 1 de 2" e "passo 2 de 2" que seria a confirmação dos dados."* Essa ideia poderia ter sido representada através da associação de um *script* mostrando a sequencia de páginas.

Adicionalmente, a anotação foi bastante utilizada para expressar o que os participantes não conseguiram fazer diretamente na página. Por exemplo, Rute escreveu: "Frase deve estar em destaque. Talvez modificar a cor, tipo de letra, etc." (ver Figura 5.1). Lembramos que não disponibilizamos operações para mudar a cor, tamanho ou tipo de letra.

Se por um lado o poder de expressão da linguagem era limitado porque estávamos restritos pela tecnologia adotada para concretizar o modelo, por outro lado a anotação ampliou a expressão do modelo.

Cunha (2001) já tinha evidenciado o uso de anotações em sua pesquisa sobre extensões de sistemas por usuários finais. Elas foram utilizadas pelos estendedores para atingir diversos objetivos: "explicar e dar feedbacks sobre a representação; descrever e explicar a extensão ou parte dela, o seu funcionamento e regras de funcionamento; justificar a criação da extensão;

Figura 5.2: Evidência do Uso de Anotação.

fazer reflexões e analogias; e documentar comunicação síncrona feita com a avaliadora/designer." (Cunha, 2001: p. 114)

5.1.8 Flexibilidade e Expressividade

Na instanciação de parte do modelo limitamos a expansão do sistema de significação da aplicação original aos "tipos" de elementos existentes na UIL da parte do sistema que está sendo modificada. Porém, novas instâncias destes elementos (*tokens*) podem ser criadas. Por exemplo, se existe o objeto "Botão ok" na interface original, é permitido criar um novo signo do tipo "botão" (e.g botão "Quero receber email"). Portanto, a expansão do sistema de significação da aplicação original se dá a partir de "tipos" da UIL. Adicionalmente, estamos aumentando um pouco mais o poder de expressão, ao permitir a representação de processos (signos dinâmicos) associados a elementos da interface (e.g associar um *script* a partir de um botão). Para os casos onde os interlocutores são impossibilitados de expressar totalmente sua modificação, estamos ainda provendo código adicional (como foi sugerido em (Cunha, 2001)) para que

eles tenham uma forma alternativa de especificar a modificação e de se fazer entender pelo receptor de sua mensagem. Isto é feito através da anotação (linguagem natural) sobre elementos da interface. Por exemplo, Rute tentou criar um novo *radio button*, mas só conseguiu copiar a bola de seleção e não conseguiu inserir o *label* do *radio button*, então escreveu a anotação: *"Inserir a opção: aguardando resposta do aluno"*. O mais importante foi que Raquel captou a ideia:

> *"... como ela botou o comentário ficou super claro... o fato de não estar o texto aqui não é importante porque já tem aqui na explicação". (Fala de Raquel)*

Além disso, o poder de expressão da Linguagem de Modificação concretizada na primoTiWIM é limitado pelo poder de expressão dos sistemas de significação da tecnologia envolvida (i.e. sistemas de significação do sistema operacional, da ferramenta de email, da arquitetura web, etc.).

Em (de Souza et al., 2001) os autores fizeram uma avaliação, usando princípios da Engenharia Semiótica, sobre ambientes de EUP. Eles já apontavam para a necessidade de uma base semiótica que fosse além dos tipos instanciados pelo *design* concreto da interface. Ou seja, a expansão dos significados deveria levar em conta elementos extraídos da "cultura computacional" dos usuários.

Porém, o que vimos nesta pesquisa é que nem sempre é interessante mudar o sistema de significação da UIL neste nível. Existem casos onde a discussão acontece com ideias maduras e sendo assim, as pessoas não fazem mudanças radicais na aplicação, como por exemplo, inserir novos tipos. Por outro lado, existem discussões sobre ideias pouco claras e neste caso é preciso haver uma atividade reflexiva. Logo as ferramentas de EUD/EUP devem permitir a escolha do nível de flexibilidade que os interlocutores poderão ter na discussão, ampliando assim, o poder de expressão e consequentemente apoiando a elaboração de alternativas numa etapa de reflexão.

5.1.9
Tipos de Modificação de Acordo com a Engenharia Semiótica

Nosso modelo permite que os participantes da discussão expressem vários tipos de modificação propostos em (de Souza, 2005) e apresentados do Capítulo 2 (seção 2.3.2). Algumas características de como ele foi concretizado mostram que a representação de algumas modificações deve atender a requisitos práticos. Por exemplo, para expressar corretamente uma alteração do tipo III (remoção de signos) a ferramenta (primoTiWIM) que implementa parte

de nosso modelo deveria destacar os signos impermeáveis e ainda ter o recurso de agrupar signos. Já para expressar o tipo VII (reprogramar), a primoTiWIM não deveria limitar a criação de novos elementos aos tipos de elementos existentes, isto causaria uma dificuldade para o usuário "reprogramar" o sistema. Por essa limitação, a representação do tipo VI (criação de estilos, por exemplo) também ficou comprometida, apesar de que um estilo existente na interface pode ser modificado sem ser criado novos tipos.

O tipo I pode ser expresso pela primoTiWIM porque ela permite o usuário mudar os nomes dos signos. Porém, como a primoTiWIM não fez distinção de signos impermeáveis, pode ocorrer de o usuário mudar o nome de um botão "salvar" para "arquivar", alterando assim a identidade da aplicação. O tipo II também foi expresso pela primoTiWIM porque ela permite mudar a sequência dos elementos de interface. Adicionalmente, o tipo IV também é permitido pela primoTIWIM porque o usuário pode inserir anotações em elementos da interface informando seu novo significado, dessa forma a intenção seria comunicada.

No Estudo Empírico II evidenciamos vários casos de "Renaming", onde a modificação afeta apenas a dimensão léxica (tipo I do ponto de vista computacional). Por exemplo, Raquel sugeriu trocar o nome do botão "Encaminhar" para "Criar Requerimento". Porém, a funcionalidade do botão é a mesma, ela só alterou o nome do objeto, ou seja, a sintaxe e a semântica continuam as mesmas.

Os estudos desta pesquisa nos chamaram atenção para o fato de que a liberdade de o usuário poder excluir elementos da página pode acarretar na remoção de signos "Essenciais". Uma sugestão seria a primoTiWIM oferecer ao designer uma opção para definir quais signos da aplicação são impermeáveis. Neste caso, quem executaria a primeira etapa do modelo seria alguém da equipe de design e neste momento esta pessoa definiria que signos não podem ser removidos do sistema. Além disso, seria interessante haver uma opção para a definição de signos acidentais, a qual permitisse criar "blocos" de signos relacionados às funcionalidades que podem ser podadas do sistema.

Outro tipo de modificação não recomendada em (de Souza, 2005) é o tipo V (alteração nas dimensões léxicas e semânticas). A primoTiWIM permitiu que este tipo de modificação acontecesse. Por exemplo, existe uma comunicação (implícita) do designer do SR que é: mensagens de "alerta" do sistema devem ser vermelhas e com letras menores do que as letras do formulário. Algum interlocutor pode não perceber isto e criar novo objeto partindo do "tipo" da mensagem de alerta , mas que será uma mensagem de erro, por exemplo.

Se por um lado a liberdade do usuário poder excluir elementos da

página deliberadamente pode causar problemas, como a remoção de signos "Essenciais", por outro, se permitimos a remoção de objetos livremente, estamos dando a possibilidade de interlocutores removerem da interface elementos que não estão em questão. Em outras palavras, a remoção de elementos é uma forma de dizer: "Não quero falar sobre isso agora." e desta maneira, concentrar a discussão na parte da interface que está em foco.

Além disso, outra vantagem de não limitarmos a remoção de elementos é quando a ideia ainda está abstrata na cabeça dos interlocutores. Por exemplo, durante o processo de negociação de alternativas de projeto, pode existir uma etapa inicial de "reflexão coletiva" na qual os participantes conjecturam possibilidades de reprojeto. Neste momento a ideia ainda não está bem definida e os interlocutores precisam ter liberdade de expressão maior que um momento onde ele quer só especificar sua ideia.

Apesar da primoTiWIM permitir estes tipos indesejados de modificação da página, observamos nos estudos empíricos que havia preocupação dos participantes em preservar a aplicação. Inclusive, as modificações foram bastante pontuais (mudaram só o que estava em foco), ou seja, não houve a intenção de "mudar tudo". Naturalmente, a metamensagem dos *designers* foi respeitada.

No contexto de "reflexão" é interessante expandir a linguagem de modificação para permitir habilitar e desabilitar a inclusão liberada de "tipos" e a remoção liberada de signos, inclusive impermeáveis. Pode-se disponibilizar um conjunto de tipos pré-estabelecidos, de acordo com a cultura dos usuários do sistema discutido. Por exemplo, se faz parte da cultura deles utilizar os tipos de signos do sistema de significação do Windows, então podemos selecionar estes tipos de signos para compor o sistema de significação da primoTiWIM. Essa seleção poderia ser feita pelo administrador do grupo.

A necessidade de inclusão de novos tipos de signos nas páginas foi claramente evidenciada nos estudos empíricos. Portanto, a linguagem de modificação adequada não é a que permite expandir o sistema de significação somente partindo de "tipos" que estão em uma ou mais páginas específicas (objetos de discussão). É melhor incluir e disponibilizar, para cada página, todos os "tipos" que ocorrem no conjunto global de páginas do sistema, pois é este conjunto que constitui a UIL do sistema e não somente os tipos que estão em uma página do sistema.

Finalizando esta seção, concluímos que, de maneira geral, a concretização de parte de nosso modelo teve uma boa aceitação e que as pessoas gostaram da ideia de utilizar a UIL como forma de expressão. Quando perguntamos a opinião sobre o modelo, a maioria dos participantes disse gostar da ideia, por

exemplo, Raquel falou: *"Achei que foi uma boa sacada sair apontando o que a gente deve fazer"*. Raquel também comentou que essa forma de comunicar é melhor ainda para o usuário final porque:

> *"Eles iam perceber menos limitações do que a gente". (Fala de Raquel)*

Vimos também que existem momentos da discussão em que a reflexão é vital, e nestes casos, é interessante que a linguagem de modificação tenha um grande poder de expressão. Já quando ideia está madura, é interessante restringir os tipos de modificação que podem ser realizados.

Adicionalmente, podemos dizer que o contexto de aplicação do modelo aqui proposto pode ser extrapolado para sistemas Web, e não somente para sistemas de grupo. Isto porque os estudos empíricos mostraram que comunicar ideias sobre modificação de sistemas através da UIL é interessante, independente de existirem diversos papéis no sistema (o que caracteriza os sistemas de grupo).

5.2 End User Development - EUD

Nesta seção fazemos uma apreciação de nosso trabalho a partir de seus correlatos na área de EUD, ressaltando seu diferencial e mostrando aspectos que ainda devem ser tratados. Iniciamos com relatos sobre as dificuldades de promover o envolvimento de usuários finais no desenvolvimento de sistemas. Em seguida, apresentamos algumas iniciativas que estão sendo feitas para romper essas dificuldades. E por fim, discutimos nossa contribuição para EUD.

O envolvimento de usuários no desenvolvimento de sistemas (EUD) tem atraído significante atenção nos últimos anos devido à necessidade dos usuários de ajustarem as aplicações de acordo com suas experiências e desejos (Cypher & Halbert, 1993). A web 2.0, por sua vez, tem tornado possível o desenvolvimento de uma nova "cultura de participação" do usuário e de "criatividade social" (Fischer, 2010).

Porém, promover EUD ainda é uma tarefa difícil. Uma destas dificuldades é devida ao fato de que a programação é uma atividade humana que requer significativo esforço cognitivo (Blackwell, 2002). Além disso, Lin e Tatar (Lin & Tatar, 2010) argumentam que a dificuldade de programar está relacionada também à questões sociológicas, pois a forma rígida de pensar utilizada na engenharia restringe as características maleáveis do design.

Outra dificuldade em EUP é apresentada por Fischer (Fischer et al., 2004): ele argumenta que o projeto de linguagens para usuários

finais revela conflitos entre complexidade e potencial. A ferramenta ideal para EUP deve reduzir a necessidade de aprendizado e fornecer recursos poderosos para resolver problemas.

Neste contexto, "um desafio fundamental para os próximos anos é o desenvolvimento de ambientes que permitam os usuários sem conhecimento de programação, desenvolver e modificar seus próprios aplicativos" (Lieberman et al., 2006). Vários trabalhos têm sido publicados neste sentido e alguns deles foram apresentados no capítulo 2. O CoScripter é um bom exemplo deste tipo de ambiente, visto que ele oferece ao usuário final a possibilidade de ele automatizar suas tarefas. Além disso, o CoScripter preocupa-se claramente em promover uma cultura de participação e de "criatividade social" uma vez que ele disponibiliza uma *Wiki* de *scripts* produzidos pelos usuários. Porém, existe um problema na disponibilização destes *scripts* porque, uma vez publicados na Wiki, as pessoas podem modificá-los sem nenhuma restrição, podendo até alterar a função para o qual um *script* foi feito. Inclusive, este é um dos problemas levantado por Fischer (2007) quando ele fala em "criatividade social": quem pode contribuir para a Wiki? Quem pode utilizar os recursos? Quais os limites de acesso?

Assim como nós utilizamos o CoScripter com uma finalidade específica, diferente da que ele se propõe, outros trabalhos também o fizeram. Um bom exemplo é o Vegemite (Lin et al., 2009), um sistema que permite combinar dados de vários sites em uma tabela (que eles denominaram de VegeTable), com um detalhe: as colunas desta tabela podem ser obtidas através da execução de um *script* do Coscripter. Além disso, este *script* pode ler campos da própria tabela para em seguida fazer uma busca na web e colocar o resultado na VegeTable. Outra ferramenta que usa o CoScripter é Highlight (Nichols & Lau, 2008), a qual permite usuários finais customizarem sistemas Web para dispositivos móveis. Os autores representaram a navegação de páginas utilizando o CoScripter. Este trabalho é bastante interessante e relaciona-se como o nosso, pois todos nós queremos oferecer ferramentas ao usuário final para representar modificações em sistemas Web (tanto processos como a aparência das páginas). Porém, existem duas principais diferenças. (i) Eles querem modificar as aplicações para melhorar a usabilidade enquanto nós queremos modificar páginas com finalidade de comunicar uma ideia que será amadurecida coletivamente. (ii) Eles focam em dispositivos móveis, cuja interação é bastante restrita e nós em sistemas Web.

Outra importante ferramenta para customizar e estender sistemas, apresentada no Capítulo 2, é a Chickenfoot (Bolin et al., 2005), a qual permite automatizar operações Web, integrar "web sites" e mudar a aparência das

páginas. A principal desvantagem desta aplicação é que seus usuários precisam conhecer HTML e JavaScript. Ou seja, o esforço cognitivo de aprendizagem para o usuário final é bastante alto, visto que eles não conhecem estas linguagens.

Todas estas tecnologias são um grande avanço na área de EUD/EUP, porém elas não apresentaram ainda comprovações sólidas de que são adequadas para o usuário final. Até porque a comprovação da sua adequação passa pela adoção em larga escala dessa tecnologia. Portanto, estamos reportando uma pesquisa onde ainda não se sabe se os usuários finais não são mais participativos porque eles não querem, ou porque eles não conseguem (já que nunca lhes foi dado um ambiente suficientemente bom para apoiar essa atividade). Nossa aposta é na ideia de que eles precisam de ferramentas de apoio. Para isso, estamos construindo um ambiente em que pessoas podem expressar coisas típicas de programação de sistemas sem usar jargões e ontologias atreladas a essa área (de programação). Estamos promovendo uma tecnologia para habilitar o usuário final a fazer isto. Se eles vão fazer, ainda não sabemos. Logo, nesta pesquisa, não estamos provando que essa tecnologia é boa para usuário final, mas estamos investigando se o usuário final sabe usar a UIL para evoluir sistemas e quais os tipos de modificação devem ser permitidas de maneira que preserve a identidade da aplicação.

Nosso trabalho pode promover a "Cultura de Participação" através da co-autoria de usuários finais em processos de desenvolvimento de software. Além disso, pode promover a "Criatividade Social", uma vez que a tarefa de evoluir sistemas é vista como um grande processo cooperativo, no qual a ideia de modificação é amadurecida por um grupo de pessoas. A promoção destes paradigmas é, em si, uma contribuição desta pesquisa, uma vez que eles dão novas oportunidades para produção social, colaboração coletiva, e até ação política e educação. (Fischer, 2010)

A participação no desenvolvimento tecnológico como forma de ação social tem precedentes. O designer participativo (Muller, 1993) é um exemplo clássico. Pesquisas sobre Design Participativo (Schuler, D. & Namioka, A., 1993) abrangem uma grande diversidade de teorias, práticas, análises e ações com o objetivo de trabalhar diretamente com os usuários do projeto, de maneira efetiva. A ideia não é somente envolvê-los superficialmente mas fazer com que eles assumam alguma tarefa na prática. Os pesquisadores defendem a ideia de que para o desenvolvimento de sistemas ser eficiente a equipe de projetistas deve trabalhar em conjunto com os conhecedores do negócio do domínio da aplicação. Um dos problemas estudados nesta área é a dificuldade de comunicação entre os envolvidos no

processo, visto que eles precisam falar sobre conceitos de "engenharia de software" de maneira que usuários compreendam. Se imaginarmos o modelo aqui proposto no contexto de desenvolvimento de sistemas, no qual a equipe de desenvolvedores utilizaria nosso modelo para discutir alternativas de projeto, poderíamos obter a grande vantagem de aumentar o número de pessoas envolvidas no processo de design. Isto porque elas teriam uma ferramenta de apoio para expressar e comunicar suas ideias de projeto ao mesmo tempo que não precisariam estar presentes fisicamente em um mesmo momento. Portanto, este trabalho pode contribuir para a expansão da fronteira do Design Participativo.

5.3 Engenharia Semiótica

Esta pesquisa é uma contribuição para a Engenharia Semiótica, uma vez que estamos oferecendo uma tecnologia para ajudar designers e co-designers a fazerem Engenharia Semiótica. Isto porque o modelo proposto (assim como sua instanciação) tem uma metacomunicação implícita, que é a de operacionalizar uma conversa sobre metacomunicação dos designers originais para os usuários finais.

Adicionalmente, a tecnologia que oferecemos permite a usuários finais usarem conceitos importantes da Engenharia Semiótica sem que eles conheçam os fundamentos desta teoria. Por exemplo, a TiWIM abre uma possibilidade de projetistas e usuários finais expressarem-se através de signos (estáticos, dinâmicos e metalinguísticos) e isso faz parte da essência da Engenharia Semiótica. Isto é inovador pelo desafio de concretizar os conceitos desta teoria e porque os estados atuais das ferramentas epistêmicas da Engenharia Semiótica ainda não conseguem introduzir seus conceitos, modelos e ontologia, na prática cotidiana.

Adicionalmente, a tecnologia que propomos permite investigar novos aspectos de como a mensagem de metacomunicação é percebida e apropriada pelo usuário. Isto é uma contribuição para futuras pesquisas teóricas sobre Engenharia Semiótica.

Vimos no Capítulo 2 que a Engenharia Semiótica apresenta dois princípios para analisar linguagens quando os significados de uma são definidos ou expressos em outra. Nosso modelo não implementa a modificação que está sendo proposta, ou seja, a linguagem para representar a modificação permite uma representação da ideia sem contudo programá-la, pois a finalidade é a "comunicação" da evolução do sistema. Portanto, o princípio "Contínuo

Semiótico" não se aplica ao modelo aqui proposto. Já o princípio de Abstração Interpretativa pode ser observado em nosso modelo, pois a semântica que impera na mente dos usuários é a semântica da linguagem de interface (UIL), ou seja, ele não precisa conhecer nenhuma linguagem de programação para expressar suas ideias de modificação no sistema. Mas quando nosso modelo permitir o usuário programar tais modificações, podemos dizer que ele está convergindo para atender aos dois princípios.

Na implementação da primoTiWIM, o gravador de macros CoScripter segue o princípio da Abstração Interpretativa ao permitir que o usuário final crie uma funcionalidade sem que ele conheça a linguagem em que ela (a nova funcionalidade) é gravada. O usuário só precisa conhecer sobre como interagir com o sistema (UIL), a partir desta interação a nova funcionalidade é gravada no código de *script* do CoScripter.

Também existe uma possibilidade de que a linguagem de *script* do CoScripter e a UIL serem semioticamente contínuas. Isto é verdade se o usuário conseguir escrever a extensão utilizando o código do Coscripter e antevendo como vai ficar o resultado na interface do sistema. Porém, nesta pesquisa não chegamos a avaliar isto.

5.4 Sistemas de Grupo

Segundo Easterbrook (1995) muitos softwares funcionam bem não porque eles foram projetados adequadamente mas porque seus usuários adaptam-se para atender suas necessidades. No contexto de sistemas de grupo a questão é: será que trabalhos colaborativos são igualmente adaptáveis? Easterbrook argumenta que projetar sistemas de grupo (*groupware*) é particularmente difícil porque não há um *framework* sólido para entender como membros de um grupo adaptam-se para lidar com a "coordenação de breakdowns" e "conflitos". O autor refere-se a "coordenação de breakdowns" como uma incompatibilidade entre expectativas e ações entre os participantes. O evento que causa *breakdown* pode ser um ato de comunicação, e neste caso, a incompatibilidade pode ser o resultado de um erro de comunicação ou de percepção entre as partes, ou ainda entendimentos diferentes da situação. O "conflito" é a interação entre pessoas que percebem ter metas, objetivos e valores opostos e que veem o outro como alguém que pode interferir em suas metas (Easterbrook, 1995).

Podemos dizer que nosso modelo se caracteriza justamente por representar e comunicar da melhor maneira possível a "intenção" de design, o que pode contribuir, no contexto de negociação de soluções de *design* de sistemas de grupo, para esclarecer o problema de entendimentos diferentes da

situação (*breakdown*).

Uma das formas de endereçar a problemática de projetar aplicações de grupo é os designers passarem a refletir cuidadosamente sobre implicações éticas e sociais do artefato que eles estão produzindo (de Souza, 2005). A Contribuição da Engenharia Semiótica para isto é prover aos designers materiais que os apoiam nesta reflexão. Nosso modelo, como um signo da Engenharia Semiótica, promove a reflexão sobre os papéis protocolados no sistema em questão (característica importante de sistemas de grupo), uma vez que os interlocutores devem comunicar sobre os papeis envolvidos na modificação. O ato de ter que comunicar é em si uma atividade epistêmica. Esta representação dos papéis na comunicação pode ser útil para resolver problemas de conflitos, uma vez que a divergência de interesses pode estar associada aos diferentes papéis do domínio que cada pessoa executa no sistema de grupo. Pode ser que a resolução do conflito seja criar uma solução de *design* para cada papel de domínio do sistema, ao invés de encontrar uma solução única para todos.

O modelo permite que interlocutores falem em nome de outros papéis. Por exemplo, a pessoa no papel de "designer" pode falar no papel de uma "secretária". Porém, vimos nos estudos empíricos que nenhum interlocutor falou em nome de outro papel. Isto é um indicador para pesquisas futuras sobre essa pessoa do discurso (primeira pessoa ou terceira pessoa) que o interlocutor prefere expressar-se.

Neste capítulo apresentamos e discutimos os resultados dos estudos empíricos, mostramos como nosso trabalho se relaciona com pesquisas de EUD e pode promover ideias como Design Participativo, Cultura de Participação e Criatividade Coletiva. Adicionalmente, apresentamos as contribuições desta pesquisa para a Engenharia Semiótica e Sistemas de Grupo.