1
Introdução

Perante as facilidades e oportunidades oferecidas pela computação para melhorar a vida humana, cada vez mais ela se faz presente em nosso cotidiano, seja por uma escolha pessoal ou de forma obrigatória. É comum utilizarmos computadores no nosso trabalho, no nosso lazer, ou mesmo nos momentos em que exercemos nossa cidadania. O hardware e o software devem estar sempre prontos para apoiar a solução dos nossos problemas, adequando-se às diferentes situações de uso.

O mercado, alimentado pelo conhecimento desenvolvido na academia, tem se empenhado em construir hardware e software que apóiem a solução dos problemas do usuário, satisfazendo suas necessidades e preferências (entretenimento, educação, medicina, pesquisa, etc.). É importante notar que as necessidades e preferências do usuário geralmente mudam mais rápido do que o tempo de vida de um sistema adquirido. Quando estas mudam, geralmente o usuário se mantém satisfeito adquirindo apenas um "novo" software adaptado às suas "novas" expectativas, ao invés de adquirir um "novo" sistema por completo (novo hardware + novo software) como ocorre em sistemas embarcados[1], por exemplo. A principal vantagem de apenas adquirir um "novo" software é a financeira, pois o usuário não tem gastos com um novo hardware, mas também é a opção que aproveita melhor o tempo de vida do hardware, poluindo menos. Este comportamento do usuário exige do mercado uma capacidade de desenvolvimento

[1] Sistemas embarcados são aqueles que já vem com os principais ou todos os softwares instalados de fábrica, não permitindo que o usuário possa substituí-los por outros softwares que se adaptem às suas necessidades. Um típico sistema embarcado amplamente utilizado hoje é um telefone celular. Por acaso algum usuário comum pode/consegue instalar no seu celular uma nova agenda mais apropriada às suas necessidades? Se ele quiser uma nova agenda, ele tem que comprar outro celular.

de software com qualidade, com preço competitivo, em prazo aceitável e que corresponda às suas expectativas.

Mesmo com toda evolução da Computação atual, o software ainda é um artefato de desenvolvimento complexo. É difícil garantir que um software tenha alta qualidade de uso (além da alta qualidade interna e funcional) em todas as situações a que ele se destina, principalmente considerando-se que ele será utilizado por qualquer usuário em potencial e em diversas circunstâncias. Apesar desta dificuldade, podemos destacar duas grandes áreas de pesquisa na Computação que investigam formas de desenvolver software com maior qualidade: a Engenharia de Software, preocupando-se principalmente com a qualidade "interna" do sistema, e a Interação Humano-Computador (IHC), concentrando-se na sua qualidade de uso.

1.1 Motivação

Dentre as estratégias de tratamento de problemas complexos, existem pelo menos duas relevantes para o desenvolvimento de software: a estratégia de divisão e conquista, e a estratégia de abstração. A estratégia de divisão e conquista propõe a divisão do problema em partes mais simples e de tratamento mais fácil. Assim, a solução do problema será a composição da solução das partes. Já a estratégia de abstração propõe desconsiderar momentaneamente alguns detalhes do problema que atrapalham o tratamento do mesmo. A própria diferenciação de Engenharia de Software e IHC é uma aplicação da estratégia de divisão e conquista: a primeira cuida das funcionalidades do sistema e a segunda cuida do uso destas funcionalidades feito pelo usuário.

Modelos são recortes da realidade com níveis apropriados de abstração (Hoover et al., 1991). Quando se usam modelos, na verdade se está dividindo o problema em perspectivas diferentes para poder "simplificá-lo", tratando mais facilmente cada uma delas. Além disto, cada perspectiva pode ser tratada em diferentes níveis de abstração, onde são endereçadas diferentes questões de design. Em outras palavras, cada modelo trata da realidade sob uma determinada perspectiva e em um determinado nível de detalhes, utilizando, portanto, as duas estratégias.

A Engenharia de Software utiliza amplamente modelos como ferramenta de auxílio no desenvolvimento de software, principalmente para projetar o sistema. Em IHC, por outro lado, a abordagem de projeto de sistemas baseado em modelos não é tão utilizada. Muitos pesquisadores de IHC consideram que representações informais como cenários ou *storyboards* são suficientes para projetar sistemas interativos nas diversas etapas de design (Carroll, 1995; 2000; Landay e Mayers, 2001; Bailey et al., 2001, Lin et al., 2002; Snyder, 2003). Todavia, alguns pesquisadores de IHC estão propondo seguir processos de design baseado em modelos (Vanderdonckt e Berquin, 1999; Paternò, 2000, Clerckx e Coninx, 2003, Paternò e Santoro, 2003). A relevância atual deste tópico de pesquisa pode ser observada, por exemplo, com a realização dos seguintes eventos:

1. *Engineering of Human-Computer Interaction*[2] (EHCI) – conferência organizada pelo IFIP WG 2.7/13.4, grupo de trabalho da IFIP que investiga a natureza, os conceitos e a construção de interfaces de usuários para sistemas computacionais. Este grupo está simultaneamente sob os comitês técnicos de Engenharia de Software e Interação Humano-Computador.
2. *Design, Verification and Specification of Interactive Systems*[3] (DSVIS) - workshop internacional anual que em 2005 teve sua décima segunda edição. Alguns dos tópicos de interesse neste ano foram: *HCI models; e.g. context, user, task, object-oriented e Model based design*.
3. *TAsk MOdels and DIAgrams for user interface design*[4] (TAMODIA) - workshop internacional que em 2005 terá a sua quarta edição. Alguns dos tópicos de interesse neste ano serão: *Cognitive task models; Task models: analytic and empirical modeling* e *diagrams for task representation: textual, graphic, animation, video*.

[2] Mais informações em: http://www.se-hci.org. Último acesso em dezembro de 2004.

[3] Mais informações sobre a última edição (em 2005) em: http://www.dsvis2005.org Último acesso em abril de 2005.

Por abordarem o desenvolvimento de software sob diferentes perspectivas, o foco dos modelos de Engenharia de Software é diferente dos modelos de IHC. Na Engenharia de Software, os modelos focam a arquitetura do sistema (por exemplo, UML e Diagrama de Fluxo de Dados). Em IHC, a maioria dos modelos foca os objetivos e as tarefas dos usuários, bem como a forma como eles interagem com o sistema para realizá-las. Vale notar que investigar as tarefas do usuário em IHC tem um foco diferente da especificação de requisitos em Engenharia de Software. IHC se preocupa mais em entender para que, por que, como e quando o usuário vai utilizar o sistema como apoio às suas atividades. Por sua vez, a Engenharia de Software tem uma preocupação maior em descobrir e entender como deve ser o sistema para apoiar as tarefas do usuário.

Uma das principais funções dos modelos no projeto de sistemas computacionais é representar as decisões de projeto de forma que possam ser facilmente compartilhadas, analisadas e discutidas pela equipe de design e de desenvolvimento. É importante observar que, enquanto o projetista utiliza modelos, ele está sendo induzido e conduzido por estes modelos a compreender melhor o produto (sistema) sendo projetado, sob uma determinada perspectiva.

1.2 Objetivo

Diante deste quadro, este trabalho pretende contribuir com abordagens de projeto de sistemas baseados em modelos na área de IHC. Mais especificamente, se concentra na modelagem da interação humano-computador, onde uma "interação" é definida como uma conversa entre o usuário e o sistema, através da qual o usuário busca alcançar seus objetivos.

Nesse contexto, em 2003, Paula propôs a MoLIC[5], uma linguagem que os designers de IHC podem utilizar para modelar a interação dos usuários com

[4] Mais informações sobre a última edição (em 2005) em: http://liihs.irit.fr/event/tamodia2005. Último acesso em abril de 2005.

[5] Em sua primeira edição, MoLIC era tratado como termo masculino ("o MoLIC"). No entanto, este uso causava confusão entre a *linguagem* MoLIC e um *modelo* representado em MoLIC. Neste trabalho optamos por nos referirmos à linguagem no feminino ("a [linguagem] MoLIC"). Isto permite que, ao nos referirmos a um modelo representado em MoLIC, a expressão

sistemas computacionais (Paula, 2003; Barbosa e Paula, 2003), seguindo a metáfora de interação como conversa. Fundamentada na teoria da engenharia semiótica (de Souza, 2005), MoLIC foi elaborada para apoiar o designer de IHC no planejamento da interação, encorajando sua reflexão sobre as estratégias de resolução de problemas que os usuários poderão seguir, isto é, estratégias que serão apoiadas pelo sistema. Neste texto, o termo designer será utilizado para designar uma pessoa ou uma equipe que realiza o projeto da interação humano-computador, desde a sua concepção até a sua concretização na interface com o usuário.

Desde sua proposta, MoLIC tem sido utilizada em estudos de caso de diferentes domínios e plataformas (por exemplo, o sistema de comunicação apresentado em Gonçalves et al., 2004) que vêm sendo desenvolvidos em períodos de tempo sobrepostos e por pessoas distintas. Além das lacunas antecipadas pela autora da MoLIC, estes estudos de caso levantaram questões quanto ao uso da linguagem que tinham ficado pendentes ou que ainda não tinha sido endereçadas na primeira edição da linguagem. Algumas destas questões deram origem a propostas de extensão à MoLIC, algumas propostas já documentadas (Coelho e Barbosa, 2003; Silva e Barbosa, 2004) e outras vêm sendo discutidas pelo grupo de pesquisa do SERG[6].

Como ainda existem questões em aberto e essas propostas de extensão não tiveram o rigor necessário para serem definitivamente incorporadas à MoLIC, surgiu a necessidade de um trabalho que enderece essas questões e organize essas e outras extensões que se façam necessárias, tomando o cuidado de manter a consistência da linguagem com os seus fundamentos teóricos. Assim, este trabalho propõe a definição da segunda edição da MoLIC para satisfazer esta necessidade prática, procurando explicitar a fundamentação dos elementos da linguagem em função da teoria que a inspirou, a engenharia semiótica (de Souza, 2005).

"modelo representado em" seja omitida, como em: "um [modelo representado em] MoLIC" ou "o [modelo representado em] MoLIC", sem causar ambigüidades como ocorria anteriormente.

[6] Semiotic Engineering Research Group (SERG) – http://www.serg.inf.puc-rio.br. Último acesso em abril de 2005.

1.3 Organização da dissertação

O capítulo 2 apresenta alguns modelos relacionados com o projeto de interface de sistemas interativos e retorna aos fundamentos da MoLIC, com o intuito de adquirirmos argumentação e visão crítica necessárias à elaboração da próxima edição da MoLIC. O capitulo 3 apresenta uma revisão da primeira edição da MoLIC levantando algumas questões quanto ao uso da linguagem ainda em aberto ou não endereçadas, e, na medida em que surgem, as responde, comparando, criticando e unificando as propostas preliminares de extensão que serão incorporadas na segunda edição da MoLIC. Como resultado destas discussões, o capítulo 4 apresenta a segunda edição da MoLIC. Caso o leitor já conheça a MoLIC ou não esteja interessado no seu processo de evolução, recomendamos pular o capítulo 3 e continuar a leitura no capítulo 4. Por fim, o capítulo 5 apresenta as contribuições deste trabalho e levanta outras questões sobre a linguagem para serem investigadas e solucionadas em edições futuras da MoLIC.