

PROJETO DE INTERFACE DO USUÁRIO COM O SUPORTE DE AMBIENTES DE DESENVOLVIMENTO BASEADOS EM MODELOS

**Karolyne OLIVEIRA (1); Bernardo LULA (1); Luiz CHAVES (2); Francisco MEDEIROS
(2); Márcia de Oliveira ALVES (2)**

(1) Universidade Federal de Campina Grande (UFCG), Departamento de Sistemas e Computação, 58.109-970, Campina Grande, Brasil, e-mail: {karolyne, lula}@dsc.ufcg.edu.br

(2) Centro Federal de Educação Tecnológica da Paraíba (CEFET-PB), João Pessoa, Paraíba, Brasil, e-mail: lucachaves@gmail.com, {petronio, marcia}@cefetpb.edu.br

RESUMO

Ambientes de desenvolvimento de interfaces do usuário baseados em modelos (MB-IDE) se apresentam como a grande alternativa para aumentar a produtividade dos desenvolvedores e projetistas de interface, além de fornecer alto grau de usabilidade às interfaces desenvolvidas. Esses ambientes têm informações organizadas em componentes com aspectos relevantes de projeto de interface, objetivos do sistema e o contexto de uso proposto. Os aspectos relevantes de projeto de interface se referem basicamente ao uso de modelos (tarefa, aplicação e apresentação), técnicas de mapeamento entre esses modelos e métodos de interação. Com relação ao objetivo do sistema, esse visa descrever qual a finalidade da aplicação. Por fim, o contexto de uso descreve todas as entidades que podem influenciar como a tarefa do usuário é realizada na futura interface. Nesse artigo serão apresentados os principais MB-IDEs, descrevendo os modelos que eles utilizam, o relacionamento entre eles e as ferramentas usadas para construção de interfaces baseadas em modelos. Salvo algumas exceções, as MB-IDEs ainda não são bem recebidas no mercado, os sistemas baseados em modelos precisam capturar informações suficientemente relevantes para prover uma assistência usável, e isso deve ser feito através de uma linguagem de especificação que retenha um positivo custo/benefício aos desenvolvedores. As MB-IDEs devem dar suporte ou ao menos acomodar a construção de interfaces tão sofisticadas quanto as feitas por outros meios que o projetista está acostumado a utilizar. Ao final desse trabalho, pretende-se apresentar os requisitos para construção de um ambiente de desenvolvimento de interfaces baseado em modelos que apóie os processos definidos em uma metodologia de concepção de interface e que possa suprir os anseios de desenvolvimento ágil, centrado no usuário, que uma facilidade de uso com os ganhos de se trabalhar com modelos.

Palavras-chave: Ambiente de Desenvolvimento Baseado em Modelos, Concepção de Interfaces do Usuário, Engenharia de Usabilidade, Ferramentas e Técnicas de Projeto de Interface, Modelo da Tarefa.

1. INTRODUÇÃO

A Engenharia de Software tem evoluído para atender critérios de qualidade cada vez mais exigentes relativos à crescente demanda por sistemas computacionais mais complexos, robustos e eficientes. Uma das principais abordagens atualmente em uso para o desenvolvimento desses sistemas é a Abordagem Baseada em Modelos (*Model-Based Approach*) que consiste em estabelecer uma arquitetura que encoraje o uso intensivo de modelos como meio de gerar, de forma automatizada, seu código. Esta abordagem explora as informações contidas nos diversos modelos envolvidos para prover geração automática ou semi-automática de código e ferramentas de assistência ao projeto para diferentes tipos de aplicação. O uso deste tipo de abordagem traz grandes benefícios, como (i) a possibilidade de fácil automatização do processo através do uso de ferramentas específicas; (ii) a consistência e reutilização; e (iii) o desenvolvimento iterativo.

O desenvolvimento de interfaces do usuário, um componente crucial do software, não escapa dessa evolução e segue, da mesma forma, a abordagem de desenvolvimento baseada em modelos (LIMBOURG; VANDERDONCKT, 2004, p. 155). Nessa abordagem, modelos são utilizados para representar formalmente um agrupamento de conceitos, estruturas de representação, e uma série de primitivas e termos que podem ser usados para explicitamente capturar as várias formas de conhecimento sobre a interface do usuário e sobre sua aplicação interativa usando abstrações apropriadas.

Os processos de desenvolvimento de interface do usuário baseado em modelos fundamentam-se, tipicamente, na criação de mapeamentos entre os elementos dos modelos da tarefa, do usuário, do domínio e da interação. Muitas técnicas têm sido desenvolvidas a fim de estabelecer uma relação entre os elementos presentes nos diversos modelos contemplados nesse processo. Como forma de dar suporte a esse tipo de desenvolvimento de interface, diversos ambientes, denominados Ambientes de Desenvolvimento Baseados em Modelos (MB-IDEs) (GRIFFITHS *et. al.*, 2001), foram criados com o objetivo de auxiliar o projetista em atividades que envolvam a construção e transformação de uma coleção de modelos.

Porém, a abordagem de desenvolvimento de interface do usuário baseada em modelos apresenta dificuldades na sua adoção por parte dos projetistas, devido a aspectos importantes citados por Myers *et al.* (2000):

- (i) o uso de modelos não está associado a uma representação visual que possibilite vislumbrar a interface final que será gerada a partir dos modelos utilizados. Essa característica faz com que a interface final do usuário seja imprevisível durante todo, ou quase todo, o seu processo de desenvolvimento;
- (ii) o uso exclusivo de modelos declarativos nas fases iniciais e intermediárias do processo implica na necessidade de se aplicar técnicas de avaliação da interface final em cima desse tipo de modelo, o que impossibilita a avaliação de aspectos visuais e de navegação simultaneamente, o que seria adiado para estágios finais do processo de desenvolvimento.

Como base no exposto, este artigo apresenta uma proposta de solução para esse problema, que objetiva minimizar a resistência dos projetistas na adoção de abordagens de desenvolvimento de interface do usuário baseadas em modelos. As dificuldades identificadas por Myers *et al.* (2000) apontam para a ausência de representações visuais preliminares da interface do usuário ao longo do seu desenvolvimento como principal causador de insatisfação (implícita ou explícita) por parte dos projetistas. A solução aqui proposta tem como base a utilização de representações visuais da interface em desenvolvimento através da aplicação de diferentes técnicas de prototipagem em diferentes etapas de um processo de desenvolvimento de interface do usuário baseado em modelos.

O restante deste artigo é organizado em 5 Seções. Na Seção 2 discute-se acerca da geração de interface do usuário. Na Seção 3 serão apresentados e analisados os principais MB-IDEs e quais as principais formas de representação dos modelos expostos acima. Na Seção 4 contém a metodologia MEDITE, base deste trabalho. Na Seção 5 será apresentado um ambiente que instancia os processos da metodologia MEDITE. Por fim, na Seção 6 serão apresentados os resultados esperados e a conclusão do artigo.

2. GERAÇÃO DE INTERFACE DO USUÁRIO

A abordagem baseada em modelos foi introduzida para identificar modelos de alto nível para possibilitar a especificação e análise de um sistema interativo através de níveis orientados à semântica em oposição às manipulações físicas que acontecem em algumas ferramentas de projeto de interface que não obedecem nenhuma especificação prévia (LIMBOURG; VANDERDONCKT, 2004, p. 155).

É possível achar na literatura que existem vários *frameworks* que viabilizam o desenvolvimento de interfaces do usuário. A princípio, a maioria das propostas oferecia métodos de geração de interfaces a partir de um modelo de domínio. Entretanto, atualmente, a maioria das abordagens dirige seu desenvolvimento com o uso de um modelo da tarefa.

2.1. Geração de interface do usuário baseada em domínio

Modelo de domínio encapsula importantes entidades de uma aplicação particular juntamente com seus atributos, métodos e relações. Elementos de um modelo de domínio possuem atributos que são relevantes para a seleção de elementos de apresentação. Exemplos desses atributos são tipos de dados, o tamanho, o mínimo e máximo valor, etc. A abordagem de geração de interface baseada no domínio produz interfaces complexas porque os usuários podem ver muitos elementos ao mesmo tempo. Além do mais, quando as tarefas do usuário não são contempladas, o diálogo entre as interfaces fica limitado, produzindo interfaces estáticas.

2.2. Geração de interface do usuário baseada na tarefa

A maioria das abordagens de desenvolvimento baseada em modelos define um modelo de diálogo por meio do uso do modelo da tarefa. Informação proveniente do modelo da tarefa é explorada para automaticamente ou interativamente derivar a estrutura de navegação de uma aplicação. Alguns MB-IDEs exploram a estrutura da informação tal como as relações temporais para gerar uma série de ativações que posteriormente será usada para gerar automaticamente o modelo de diálogo e os *widgets* do modelo de apresentação. O projeto de interface baseado na tarefa como oposição ao modelo de domínio incorpora informações relativas à forma como as tarefas do usuário vão ser acionadas nas interfaces do usuário tal como os relacionamento entre essas tarefas. Esse tipo de informação fornece aspectos de usabilidade como sobreposição de interfaces, agrupamento da apresentação, etc.

2.3. Prototipagem de Interfaces do Usuário

A especificação do modelo abstrato da interação apresenta os elementos e organização de que uma interface do usuário precisa para dar suporte às tarefas identificadas no modelo da tarefa, sem levar em consideração aspectos estéticos e sobre a forma de execução. Modelo abstrato da interação também pode ser chamado de interface abstrata desde que, na sua representação (na abstração), seus elementos e a forma como esses elementos são organizados nos contextos de interação sejam apresentados de forma similar aos contextos com os quais os usuários irão interagir com o sistema. A especificação do modelo concreto da interação, denominado interface concreta, define os elementos interativos relativos a uma determinada plataforma, os aspectos estéticos (padrão, fonte, cor, tamanhos de botões, etc.) e os componentes de navegação.

Algumas das principais desvantagens do desenvolvimento de interfaces baseado em modelos tem sido a imprevisibilidade do resultado final e a falta de técnicas para a avaliação da interface final considerando uma série de modelos declarativos (MYERS *et al.*, 2000). Para superar essas desvantagens, e algumas outras, diferentes técnicas têm sido introduzidas. Uma dessas técnicas introduzidas é a prototipação de interfaces do usuário. Prototipação consiste na criação de uma versão preliminar da futura interface (protótipo), assim, o usuário e o especialista podem identificar possíveis falhas no projeto da interface do usuário, tanto do ponto de vista da parte funcional como do ponto de vista da usabilidade. Técnicas de prototipação estão definidas em três categorias principais:

- (i) Técnicas *Lo-fi* (baixa fidelidade): essa família de técnicas é a mais usada no estágio de análise de requisitos para validar os requisitos com o usuário em abordagens centradas no usuário (CONSTANTINE *et al.*, 1999). Sua construção normalmente se dá através da técnica de desenho a mão livre utilizando ferramentas simples como lápis, papel e material de escritório.
- (ii) Técnicas *Me-fi* (média-fidelidade): consistem na implementação computadorizada de uma aplicação limitada funcionalmente, contendo apenas as funções essenciais para avaliar alguns cenários específicos. As características dos protótipos de média-fidelidade consistem na união das características positivas dos protótipos de baixa-fidelidade (fácil e rápido de construir e editar) e dos de alta-fidelidade (simulação, reuso e teste de usabilidade).
- (iii) Técnicas *Hi-fi* (alta fidelidade): essas técnicas são usadas para criação de versões preliminares de uma interface do usuário com um aceitável grau de qualidade. Esses tipos de técnicas produzem um

protótipo de interface do usuário que é próximo à futura interface final. Esses protótipos são representações executáveis (código), construídos com o uso de uma linguagem de programação (ou ferramentas de apoio) e contêm as principais funcionalidades presentes na interface do futuro sistema. Eles definem, claramente, os aspectos estéticos (padrão, fonte, cor, tamanhos de botões, etc.) e os componentes de navegação. Esse tipo de protótipo está presente nas diversas metodologias de concepção de interface baseada em modelos como último artefato a ser construído antes da interface final.

3. AMBIENTES DE DESENVOLVIMENTO DE INTERFACE BASEADO EM MODELOS

Para apresentar os ambientes existentes é preciso que todos se encaixem em uma mesma referência, alguns esforços representativos e significantes feitos nos ambientes existentes, que suportam a abordagem baseada em modelos, foram selecionados. Esses esforços estão apresentados de acordo com um framework de referência que representa os vários níveis de abstração e modelos que podem ser utilizados em um processo de desenvolvimento de interface do usuário. O desenvolvimento de interfaces do usuário baseado em modelos e na tarefa do usuário está estruturado basicamente em quatro níveis de abstração definidos pelo framework de referência unificado Camaleon (CALVARY *et al.* 2003), como mostra a Figura 1. Este framework agrupa os modelos e os processos de projeto de interface até então difundidos na literatura e dá suporte ao desenvolvimento de interfaces para múltiplas plataformas tanto em tempo de projeto como em tempo de execução, apresentando-se assim como um marco teórico. É dividido em três partes:

- *Modelos ontológicos para interfaces para múltiplas plataformas*, divididos em modelos de arquétipo, que servem como entrada do processo de projeto, e os modelos observados, que são modelos efetivos que guiam o processo de adaptação em tempo de execução;
- *Processo de desenvolvimento*, que explica como produzir uma Interface do Usuário para cada contexto de uso típico. Engloba os processos de Conceitos e Modelo da Tarefa; Interface do Usuário Abstrata (AUI), Interface do Usuário Concreta (CUI), Interface do Usuário Final (FUI);
- *Processo de Execução*, que mostra como a Interface do Usuário e a estrutura do *Run-Time* podem cooperar para o objetivo em outro contexto de uso.

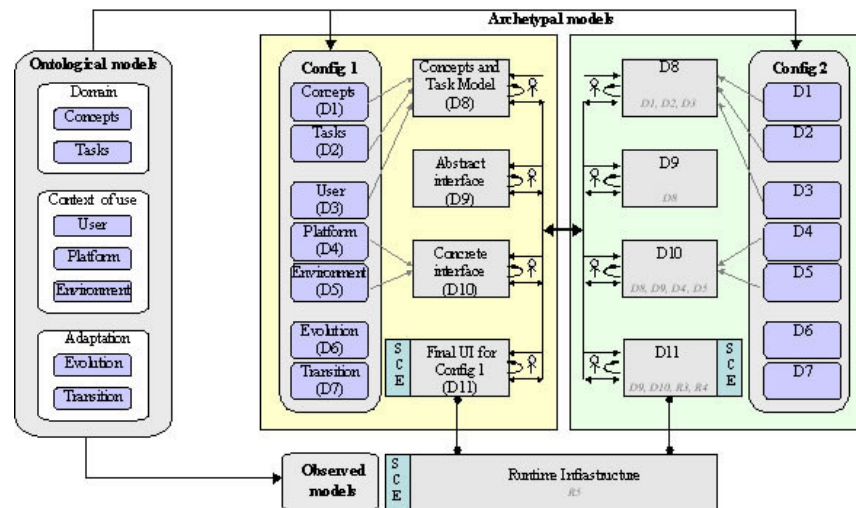


Figura 1. Framework de Referência Unificado Camaleon

Atualmente, o projeto de concepção de interfaces do usuário baseado em modelos é assistido por uma série de MB-IDEs. As primeiras, como UIDE (FORLEY *et al.* 1989) e HUMANOID (SZEKELY *et al.*, 1992) incorporaram modelo de aplicação simples, um modelo de diálogo e um modelo de interação (elementos de interface do usuário), além de um gerador de interface baseado em regras predefinidas (GRIFFITHS *et al.*, 2001). A maior contribuição de HUMANOID encontra-se no modelo de interação, ferramenta de modelagem e no desempenho, já os maiores benefícios de UIDE encontra-se no modelo de diálogo, projeto crítico e uma ferramenta de geração de ajuda.

MASTERMIND (SZEKELY, 1996) é o produto da combinação dos ambientes UIDE e HUMANOID, foi desenvolvido para capturar o que havia de melhor nos dois ambientes. Permite a geração da interface a partir de especificação da aplicação em alto nível, e para isto, requer que os desenvolvedores explicitamente especifiquem todos os níveis do modelo, desta forma, não oferece ferramenta de geração ou mapeamento automático de um modelo pra outro, o que ele disponibiliza é a capacidade de reuso das especificações e geração automática de código da interface na linguagem C++. A linguagem de modelagem permite definir o modelo de aplicação, o modelo da tarefa e o modelo da interação. Em termos do framework de referência Camaleon, MASTERMIND suporta edição dos níveis: "Tarefas e Conceitos", nível de AUI, nível de CUI e geração da FUI.

TADEUS (STARY, 2000) é simultaneamente uma metodologia e um suporte computacional ao desenvolvimento de interfaces, disponível para especificação e prototipação contextual. As atividades básicas de projeto em TADEUS iniciam-se com a especificação do modelo tarefa e do usuário de acordo com a organização do trabalho que se deseja realizar na aplicação. A partir desta, o modelo de dados é derivado, seguido da migração para elementos de interação, que já apresenta resultados prévios. Finalmente, pode ser feita tanto à prototipagem com os usuários finais quanto a geração de código a partir de uma representação orientada a objetos. Este ambiente suporta as seguintes modelagens: tarefa, usuário, dados, interação e aplicação. TADEUS permite aos projetistas estabelecer diferentes tipos de relacionamentos, mas não existem manipulações na ferramenta para explorar a semântica desses relacionamentos de modo a derivar um novo modelo, ou seja, não existe geração automática de modelos. Em termos do framework de referência Camaleon, os editores de TADEUS suportam edição dos níveis: "Tarefas e Conceitos", nível de CUI e geração da FUI.

Teallach (GRIFFITHS *et. al.*, 2001) é um ambiente que auxilia o projetista no desenvolvimento de interface que faça ligação com um banco de dados. No ambiente é possível fazer a modelagem da tarefa, do domínio e da interação. O Modelo da Tarefa requer um significado de criação e invocação de funcionalidades específicas como sessão, transação e query. O Modelo de Domínio é obtido através da análise e conversão do esquema do banco em um modelo de domínio utilizando blocos de construção. O Modelo de Interação não contempla as diversas janelas do sistema, apenas as que persistem dados. Quanto ao intercâmbio entre os modelos, existem duas maneiras de associar elementos de diferentes modelos, fazendo: (i) Ligação: É feita uma associação entre os atributos existentes nos dois modelos, ou, (ii) Derivação: Componentes de um modelo são construídos baseados nos componentes de outro. Da mesma forma, se faz uma associação entre o gerado e o gerador. Existem ligações: entre o Modelo de Domínio e da Tarefa; Da Interação para a Tarefa; Da Tarefa para a Interação; e derivação do Domínio para a Interação (itens do domínio podem ser usados para gerar itens de apresentação diretamente ou fazendo a ligação através do Modelo da Tarefa). Em termos do framework de referência Camaleon, Teallach suporta edição dos níveis: "Tarefas e Conceitos", nível de AUI, nível de CUI e geração da FUI.

TransformiXML (LIMBOURG; VANDERDONCKT, 2004, p. 155) é um ambiente que expressa qualquer tipo de mapeamento estrutural (*mapping problem*) por um suporte a expressões matemáticas dos relacionamentos (baseado numa gramática gráfica) e permite a definição e a aplicação de regras de transformação. Manipula qualquer modelo de descrição de interface do usuário expressada em UsiXML (User Interface eXtensible Markup Language) [LV04]. Seus componentes são: Modelo da Tarefa, Modelo de Interface Abstrata, Modelo de Interface Concreta e Métodos de obtenção da Interface Final. Sua arquitetura é composta de Tarefas e Conceitos; Interface do Usuário Abstrata; Interface do Usuário Concreta; e Interface do Usuário Final. O ambiente é dividido em dois componentes: uma Interface de Programação da Aplicação (TransformiXML API), que pode ser utilizado por qualquer aplicação para fornecer regras de transformação, e uma Interface do Usuário Gráfica (GUI) que serve como uma aplicação front-end para a API. Em termos do framework de referência Camaleon, TransformiXML suporta edição dos níveis: "Tarefas e Conceitos", nível de AUI, nível de CUI e geração da FUI.

TERESA (MORI *et. al.*, 2004) foi concebido para dar suporte à concepção de interfaces de sistemas para múltiplas plataformas, convertendo-os em aplicações nômades. Para garantir usabilidade, a aplicação se adapta aos diferentes contextos do usuário, em particular, aos diferentes dispositivos usados para acessar suas funcionalidades. Calvary *et al.*(2003) tem usado o termo "plasticidade" para indicar esse tipo de interface do usuário. É proposto o uso de "transcoding", que tem como objetivo fazer transformações automáticas para linguagens de diferentes plataformas a partir da especificação da interface no nível CUI. Os componentes utilizados são: Modelo da Tarefa, Modelo de Interface Abstrata, Modelo de Interface Concreta e Regras de Geração para Interface Final. O início do desenvolvimento da interface começa com

um alto nível de modelagem da tarefa para uma aplicação multi-plataforma, nessa fase, o projetista desenvolve um modelo simples que remeta aos contextos de uso possíveis e às varias plataformas envolvidas. A próxima etapa é o desenvolvimento de um modelo de tarefa para as diferentes plataformas consideradas. Nesse momento, o projetista filtra as tarefas do modelo da tarefa nômade de acordo com a plataforma que se deseja alcançar. Essa abordagem é denominada de "Abordagem Um modelo, Muitas Interfaces". Em termos do framework de referência Camaleon, TERESA suporta edição dos níveis: "Tarefas e Conceitos", nível de AUI, nível de CUI e geração da FUI.

A Tabela 1 apresenta uma visão dos MB-IDES sob o prisma do framework Camaleon.

Tabela 1. Visão dos MB-IDES pelo prisma do framework Camaleon

	Conceitos e Modelo da Tarefa	Interface Abstrata	Interface Concreta	Interface Final
Mastermind	Modelo da Tarefa Hierárquico	Descritores	Protótipo Low-Fi	Código C++
TADEUS	TATAR	Script	Protótipo Hi-Fi	-
Teallach	Modelo da Tarefa Hierárquico	Script	Protótipo Hi-Fi	Código Java Swing
TransformiXML	CCT	Representação visual rudimentar (não editável)	Protótipo Hi-Fi	XHTML
Teresa	CCT	Descritores Lógicos Múltiplos	Protótipo Hi-Fi	HTML

A maioria dos ambientes descritos oferece um método árduo de desenvolvimento para o projetista. Percebe-se, observando a tabela, que os MB-IDEs tardam a utilizar técnicas de prototipagem em seus processos e preconizam apenas o uso de um tipo de fidelidade. Além do mais, esforço para modificações das interfaces construídas através desses ambientes ainda é grande, para modificação na interface abstrata, base para geração da interface concreta, é preciso conhecimento sobre sintaxe e atributos de seus descritores, ou sobre scripts que contêm informações da interface. Da mesma forma, qualquer modificação feita em um artefato, em um determinado processo, não é refletida nos artefatos de outros processos. Uma forma de amenizar esse problema é o uso de representações visuais dos descritores em forma de protótipos de interface, que poderiam ser simulados e editados pelo projetista de modo a torná-los mais fiéis ao desejo do usuário, e que essas modificações sejam refletidas em todos os artefatos que tenham ligações com eles.

Essa abordagem trás ganho no sentido de ser bem mais intuitivo e também de antecipar a avaliação do usuário, verificando se a visualização atende os seus anseios de interação (AGUIAR, 2006). A modificação da interface nessa etapa é bem menos onerosa do que a modificação em estágios mais avançados do projeto, uma vez que a edição acontece em representações visuais ainda inacabadas. Adicionalmente, as ferramentas não devem impor um método inflexível de trabalho com uma estrutura rígida com abordagem *top-down* que só permite a construção de protótipos no término do processo de projeto (GRIFFITHS *et. al.*, 2001).

4. MEDITE

MEDITE (RODRIGUES *et. al.*, 2005) é uma metodologia baseada na tarefa do usuário, orientada a modelos, iterativa e incremental para auxiliar projetistas no processo de concepção de interfaces do usuário. O processo de desenvolvimento definido por MEDITE inclui a geração de protótipos de média- e de alta-fidelidade para representação visual da AUI e CUI, respectivamente, conforme *Camaleon*, e sugere o uso de protótipos de baixa-fidelidade como ferramenta de apoio para o levantamento de requisitos. Seu fluxo pode ser observado na Figura 2, onde as circunferências descrevem os processos utilizados e os retângulos os artefatos gerados. A metodologia em questão define quatro fases.

A *fase 1* (Tarefa&Conceito - nível 1 em *Camaleon*) define o processo de *análise e modelagem da tarefa do usuário* e para tanto utiliza o formalismo TAOS (*Task and Action Oriented System*) (MEDEIROS *et. al.*, 2000). Esta fase, pode/deve ser precedida por reuniões entre os projetistas e os clientes/usuário apoiadas pelo uso de protótipos de baixa-fidelidade como mecanismo auxiliar no levantamento de requisitos. A fase 1 tem o suporte computacional da ferramenta iTAOS (*interface TAOS*) (MEDEIROS *et. al.*, 2002) que

permite representar a tarefa do usuário a partir de uma árvore hierárquica de Tarefas, Sub-tarefas e Ações. A saída de iTAOS é um arquivo XML com a descrição da tarefa do usuário segundo o formalismo TAOS.

A fase 2 (AUI - nível 2 em *Camaleon*) define dois processos, a saber: processo de *geração da especificação conceitual parcial da interação (modelo da interação)*, onde os elementos do modelo de tarefa são mapeados em elementos do modelo da interação (SUÁREZ *et al.*, 2004). Esta fase possui como suporte computacional algoritmos de mapeamento, MAPA (*MAPPING Algorithms*) (RODRIGUES *et al.*, 2005), responsáveis por correlacionar os elementos do modelo da tarefa (Tarefas, Sub-tarefas e Ações) com os elementos do modelo da interação (Espaços, Visões e Objetos de Interação). A saída de MAPA é um segundo arquivo XML contendo uma descrição parcial da interação segundo o modelo EDITOR Estendido (RODRIGUES *et al.*, 2005). Essa descrição contém, além dos elementos de apresentação (Objetos de Interação, Visões e Espaços), a navegação (diálogo/interação) entre os elementos de apresentação (*Statecharts*), também extraídos da descrição da tarefa. O segundo processo, de *geração do protótipo de média-fidelidade da interface do usuário*, consiste na representação visual da especificação conceitual parcial da interação (AUI em *Cameleon*). Esse processo está associado ao suporte computacional de SMILE (*Sketch Manipulation Integrated with Less Effort*), uma ferramenta de geração automática, edição e simulação de protótipos de média-fidelidade. A saída desse processo é o protótipo de média-fidelidade da interface do usuário.

A fase 3 (CUI – nível 3 em *Cameleon*) também é composta por dois processos, a saber: processo de *geração da especificação conceitual total da interação (modelo da interação)* e o processo de *geração do protótipo de alta-fidelidade da interface do usuário*. O processo de geração da especificação conceitual total da interação consiste no refinamento da especificação conceitual parcial da interação a partir da adição de atributos que detêm informações extraídas de: regras ergonômicas, heurísticas de projeto, experiência dos projetistas, perfil do usuário e padrões de interface, entre outros. Esse processo em MEDITE é realizado atualmente com o auxílio de uma base de regras ergonômicas definidas por Guerreiro e Lula Jr. (2002), no entanto, sem suporte computacional. A saída desse processo é o arquivo XML da fase anterior modificado por instanciação e adições de atributos que completam a descrição conceitual da interação. O segundo processo, de geração do protótipo da interface do usuário, consiste na representação visual da especificação conceitual total da interação (CUI em *Camaleon*). Esse processo está associado à Hi-Fy, uma ferramenta de construção automática, edição e simulação de protótipos de alta-fidelidade. A ferramenta Hi-Fy encontra-se em fase de desenvolvimento. A saída desse processo é o protótipo de alta-fidelidade da interface do usuário.

A fase 4 de MEDITE (FUI – nível 4 em *Camaleon*) consiste na implementação real da interface do usuário do sistema em desenvolvimento e tem como ponto de partida o protótipo de alta-fidelidade da interface do usuário definido na fase anterior. Esta fase será assistida por algoritmos de geração de código de interface que levará em consideração tudo que foi definido na fase de edição simulação do protótipo de alta-fidelidade. A saída desta fase consiste na interface final do usuário.

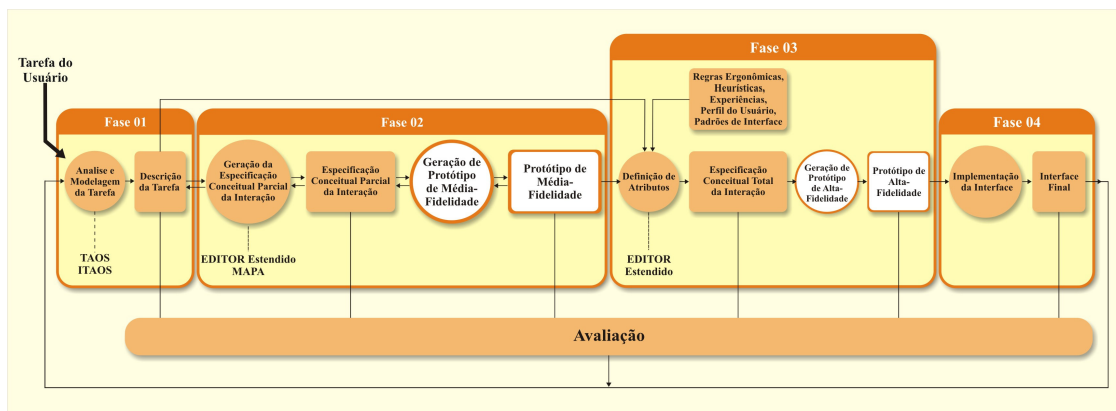


Figura 2 - Fluxo de MEDITE

Como foi mencionado, MEDITE pressupõe a utilização de quatro ferramentas de suporte computacional aos seus processos, quais sejam: (i) *iTAOS* para modelagem da tarefa; (ii) *MAPA* para geração da especificação conceitual parcial da interação (modelo da interação); (iii) *SMILE* para geração automática, edição e simulação de protótipos de média-fidelidade; e por fim, (iv) *Hy-Fy* para geração automática, edição e

simulação de protótipos de alta-fidelidade. Essas ferramentas devem funcionar integradas em um ambiente denominado *FastInterface*.

5. FASTINTERFACE

FastInterface é um ambiente de desenvolvimento de interface do usuário baseado em modelos e utilizando múltipla prototipagem que está sendo desenvolvido a partir dos seguintes requisitos básicos: (i) dar suporte ao desenvolvimento de um protótipo de baixa-fidelidade como passo inicial de levantamento de requisitos e entendimento do domínio do problema para a construção do modelo da tarefa; (ii) dar suporte à análise e modelagem da tarefa através da incorporação e disponibilização da ferramenta iTAOS; (iii) implementar através da incorporação da ferramenta MAPA os mapeamentos entre os elementos do modelo da tarefa e os elementos do modelo da interação; (iv) permitir a edição e simulação do modelo da interação a partir da edição e simulação de protótipos de média-fidelidade através da incorporação da ferramenta SMILE; (v) dar suporte à geração e edição de protótipos de alta-fidelidade da interface através da incorporação da ferramenta Hi-Fy; (vi) dar suporte à geração do código da interface final; e, por fim, (vii) oferecer a manutenção da consistência entre os modelos envolvidos na concepção da interface do usuário em tempo real de projeto.

Assim, o uso do ambiente *FastInterface* deverá favorecer a adoção da abordagem de desenvolvimento baseada em modelos, em especial o uso efetivo de MEDITE, pois os projetistas passarão a ter todo o processo de concepção da interface do usuário suportado por um único ambiente que abrange todas as ferramentas de suporte às fases existentes em MEDITE, e usufruindo das vantagens do uso evolutivo de múltiplos protótipos. A seguir, cada módulo de *FastInterface* será descrito com maior clareza de detalhes.

5.1. iTAOS

A ferramenta iTAOS (*interface* TAOS) permite a descrição dos objetivos dos usuários e dos meios utilizados no processo de realização de uma determinada tarefa, segundo o formalismo TAOS. A tarefa deve ser descrita em termos de objetivos, procedimentos, objetos, decomposição em sub-tarefas, restrições, etc. iTAOS propõe um modelo de representação hierárquico de tarefas e sub-tarefas que considera tanto o comportamento estático quanto o comportamento dinâmico de um dado domínio através de dois tipos de entidades ou conceitos: os conceitos estáticos (objetos, métodos e situações) e os conceitos dinâmicos (processos, ações e planos). Os conceitos estáticos representam entidades que não mudam de estado durante um intervalo de tempo considerável, ao contrário dos conceitos dinâmicos, que podem sofrer mudanças em um determinado intervalo de tempo.

5.2. MAPA

MAPping Algorithm (MAPA) é uma ferramenta que mapeia os elementos presentes no modelo da tarefa em elementos do modelo da interação (especificação conceitual parcial da interação), passando pelo modelo de roteiro, segundo o Modelo Editor Estendido que contempla aspectos de apresentação e de navegação/diálogo. MAPA dispõe de um mecanismo de rastreabilidade entre os modelos da tarefa e da interação que permite que mudanças realizadas no modelo da interação sejam automaticamente refletidas no modelo da tarefa e vice-versa, mantendo os modelos sempre coerentes. Para que a manutenção da coerência seja possível, um conjunto de regras descritas por (RODRIGUES *et. al.* 2005) devem ser aplicadas aos modelos. A Figura 1 ilustra o mecanismo para a manutenção da coerência entre os elementos dos modelos envolvidos.

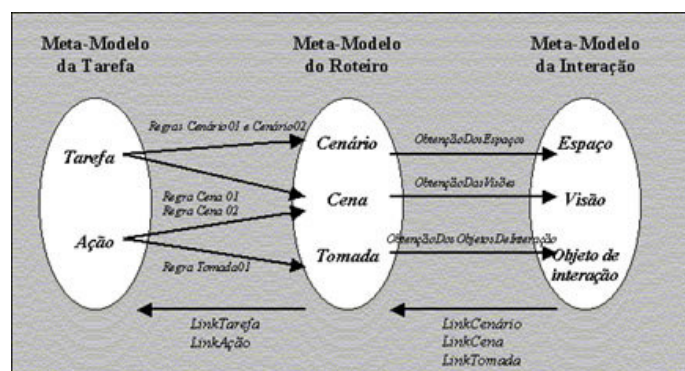


Figura 1 – Rastreabilidade entre os elementos do modelo da tarefa e do modelo de interação

MAPA recebe como entrada o arquivo XML com a descrição do modelo da tarefa obtido com o uso de iTAOS, e fornece como saída um outro arquivo XML com a descrição da especificação conceitual parcial da interação (modelo da interação). A descrição do modelo da interação fornecida por MAPA faz referência a Objetos de Interação (OIs), Visões e Espaços, onde: Objetos de Interação são quaisquer elementos que possibilitam uma interação direta do usuário com o sistema; Visões consistem em superfícies de restituição nas quais Objetos de Interação são agrupados de acordo com um contexto específico; e Espaços constituem a maior área a ser visualizada pelo usuário e são compostos por uma ou várias Visões.

5.3. SMILE

SMILE (*Sketch Manipulation Integrated with Less Effort*) é uma ferramenta computacional para o uso de protótipos de média-fidelidade como representação visual da AUI (em *Camaleon*) considerando o modelo da interação de acordo com MEDITE. Por se tratar de uma ferramenta de geração e uso de protótipos de média-fidelidade, SMILE fundamenta-se em algumas características essenciais para ferramentas desta natureza, a saber: (i) rapidez e facilidade de construir e modificar o protótipo com baixo investimento de tempo e recurso; (ii) ausência da necessidade de habilidade técnica específica por parte dos projetistas; (iii) possibilidade de explorar diferentes alternativas de projeto; (iv) melhoria na comunicação da equipe de projeto; (v) interação direta entre o usuário e o sistema; (vi) manutenção do histórico do projeto; (vii) reuso de partes do projeto; e, por fim, (viii) possibilidade de realização de testes de usabilidade e de treinamento.

SMILE recebe como entrada o arquivo XML com a descrição do modelo da interação gerado por MAPA e o apresenta visualmente sob os modos de *Árvore de Esboços* e *Esboço* corrente. As principais funções de SMILE são: (i) Inserir OIs, Visões e Espaços; (ii) Excluir OIs, Visões e Espaços; (iii) Mudar tipo de OIs; (iv) Editar Esboço; (v) Exportar Visualização; (vi) Navegar; e (vii) Simular. SMILE disponibiliza, ainda, uma função automática para a manutenção do histórico do *design* que realiza salvamento do estado atual do esboço enquanto o usuário o manipula através da geração de arquivos de *backup*. A Figura 2 apresenta uma tela de SMILE durante a edição do protótipo de média-fidelidade.

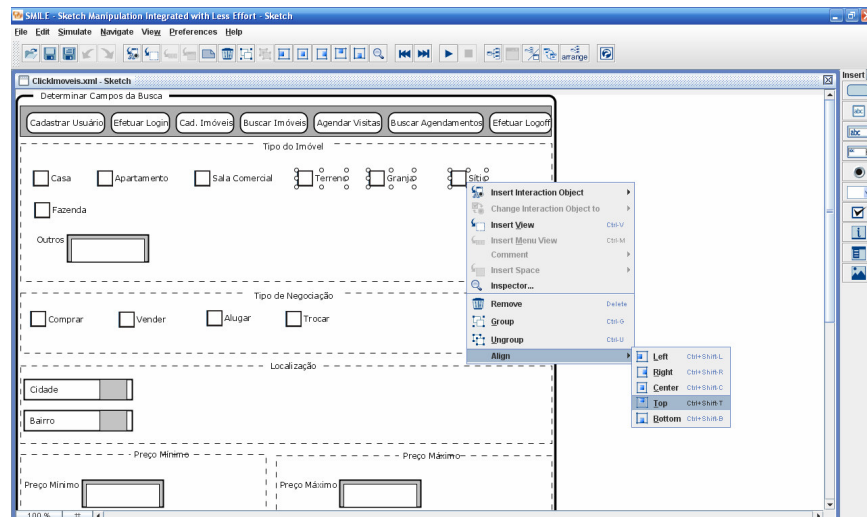
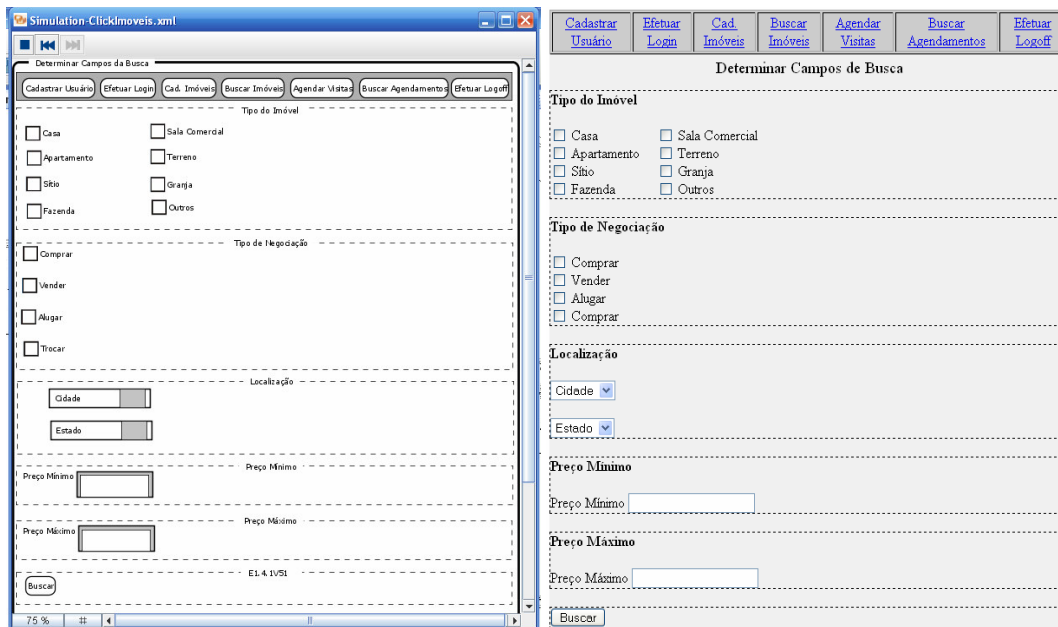


Figura 2 – Tela da ferramenta SMILE

5.4. Hi-Fy

Hi-fy é uma ferramenta computacional para o uso de representação visual da CUI (em *Camaleon*) como um protótipo de alta-fidelidade a partir da descrição da interação fornecida por SMILE. Hi-fy recebe como entrada o arquivo XML que representa o protótipo de média-fidelidade gerado e manipulado em SMILE e mapeia seus elementos em elementos gráficos específicos de uma determinada plataforma obtendo um protótipo de alta-fidelidade. Hi-fy aplica um estilo padrão a este protótipo, o apresenta visualmente e, em seguida, possibilita sua transformação em código da interface final. O protótipo de alta-fidelidade pode ser editado de modo a deixar suas representações mais próximas ao que o usuário almeja.

A Figura 3 apresenta o protótipo de média-fidelidade gerado por SMILE (a) e o protótipo de alta-fidelidade produzido por Hi-Fy (b).



(a) – Protótipo de média fidelidade

(b) – Protótipo de alta fidelidade

Figura 3 - Tela representação do protótipo de alta fidelidade levando em consideração informações provenientes do protótipo de média fidelidade

6. CONCLUSÃO

Neste artigo procurou-se mostrar os ganhos significativos na concepção de interface com o uso de Ambientes de Desenvolvimento Baseados em Modelos. Porém mesmo, com os ganhos de se trabalhar com modelos, os MB-IDEs ainda apresentam algumas limitações. Observa-se nas MB-IDEs apresentadas algumas dessas limitações, a saber: (i) MASTERMIND e TADEUS não oferecem ferramentas de geração ou mapeamento automático entre os diferentes modelos, (ii) Teallach não contempla o projeto das diversas janelas do sistema, apenas as que persistem dados, (iv) TransformiXML não permite edição de protótipos em etapas iniciais do projeto de interface e (v) TERESA não prevê o uso de representações visuais em etapas iniciais do processo de concepção. Assim, uma proposta de desenvolvimento de interface baseada em modelos associada à utilização de múltiplos protótipos dentro de um processo evolutivo com diferentes níveis de abstração foi apresentada.

A disponibilização de um ambiente único e integrado de desenvolvimento de interfaces atendendo aos requisitos apresentados acima deve proporcionar: (i) benefícios com relação à produtividade e custo de projeto de interface; (ii) menor esforço cognitivo ao projetista por utilizar uma infra-estrutura mais próxima à sua realidade de desenvolvimento, que alia facilidade de uso com o poder do uso de modelos; (iii) suporte computacional aos processos de MEDITE em um ambiente único e integrado; (iv) a manutenção da consistência entre os diversos modelos previstos na metodologia MEDITE seja realizada em tempo real de projeto; (v) o fomento à inserção de novos modelos ao ambiente para que o usuário tenha mais flexibilidade em escolher o tipo de formalismo deseja utilizar. Um ambiente, denominado *FastInterface*, que tem como objetivo dar suporte computacional e agilizar a construção de interfaces com o auxílio dessa abordagem está sendo construído. O ambiente vai congrega diferentes ferramentas de modo a oferecer uma construção evolutiva dos diferentes tipos de protótipos considerados na literatura. Esta proposta está na confluência dos trabalhos apresentados por Montero e López-Jaquero (2006) sobre o ambiente IDEALXML e por Coyette *et al.* (2007) sobre multi-fidelidade.

REFERÊNCIAS

AGUIAR, Y. P. Smile: uma ferramenta de apoio ao uso de esboço em medite. **WDCopin**, 2006.

CALVARY, G. ; COUTAZ, J. ; THEVENIN, *et al.*. A Unifying Reference Framework for Multi-Target User Interfaces. In **Interacting with Computer**, p. 289–308, 2003.

CONSTANTINE, L. L., LOCKWOOD, L. A. D. Software for use. **Addison-Wesley**. 1999.

COYETTE, A.; KIEFFER, S.; VANDERDONCKT, J. Multi-fidelity Prototyping of User Interfaces, in Proc. of 11th IFIP TC 13 Int. Conf. on Human-Computer Interaction **INTERACT'2007** (Rio de Janeiro, 10-14 September 2007), Lecture Notes in Computer Science, Vol. 4662, Springer-Verlag, 2007, pp. 149-162.

FOLEY, J; KIM, W.C ; KOVACEVIC, S.; and K. MURRAY. Defining interfaces at a high level of abstraction. **IEEE Computer**, 1989.

GRIFFITHS, T; MCKIRDY, J; PATON, N. W. ; GRAY, P. D.; KENNEDY, J.; COOPER, R.; GOBLE, C. A.; WEST, A. and SMYTH, M.. Teallach: A model-based user interface development environment for object databases. P. 31-68. **Interacting with Computers** 14, 1, 2001.

LIMBOURG, Q.; VANDERDONCKT, J. Addressing the Mapping Problem in User Interface Design with UsiXML. In **TAMODIA 2004**, p. 155-163, 2004.

MEDEIROS, H.; KAFURE, I.; LULA JR, B. TAOS: a task-and-action oriented framework for user's task analysis in the context of human-computer interfaces design. In: **XX International Conference of the Chilean Computer Science Society (SCCC'2000)**, Santiago de Chile, Chile, 2000.

MEDEIROS, HAMURABI; KAFURE, IVETTE; LULA, BERNARDO JR. TAOS: a task-and-action oriented framework for user's task analysis in the context of human-computer interfaces design. In: **XX International Conference of the Chilean Computer Science Society (SCCC'2000)**, Santiago de Chile, Chile, 2000.

MONTERO, F.; LÓPEZ-JAQUERO, V., IdealXML: An Interaction Design Tool-A Task-Based Approach to User Interfaces Design, Proc. of 6th Int. **Conf. on Computer-Aided Design of User Interfaces CADUI'2006** (Bucharest, 6-8 June 2006), Chapter 20, Springer-Verlag, Berlin, 2006, pp. 245-252.

MORI, G.; PATERNÒ, F; SANTORO, C. Design and development of multidevice user interfaces through multiple logical descriptions. **IEEE Transactions on Software Engeneering**, 30(8), 2004.

MYERS, B.; HUDSON, S. E. and PAUSCH, R. Past, present, and future of user interface software tools. **ACM Transaction. Comput.-Hum. Interact.** 7, 1 (Mar. 2000), 3-28, 2000.

RODRIGUES, C. E. C. L. Medite+: utilizando o processo de roteirização para a obtenção do modelo de interação Editor-estendido. **Dissertação (Mestrado em Informática) – Ciência da Computação, Universidade Federal de Campina Grande**, Campina Grande PB, Junho de 2005.

RODRIGUES, C. E. C. L, Lula, B, Suárez, P. R., Using a script model to preserve the consistency within an UI design environment. **Proceedings of the 4th international workshop on Task models and diagrams**, 2005.

STARY, C. Contextual prototyping of user interface. In DIS, Brooklyn, New York, 2000.

SZEKELY, P. Retrospective and challenges for model-based interface development, 1996.

SZEKELY; LUO, P.; and NECHES, R. Facilitating the exploration of interface design alternatives: The humanoid model of interface design. 1992.

SUÀREZ, P. R., LULA, B., BARROS, M. A. Applying knowledge management in UI design process. Proceedings of the 3rd annual conference on Task models and diagrams, **ACM International Conference Proceeding Series**; Vol. 86, 2004.