6 Conclusões

6.1.Trabalhos Relacionados

O primeiro trabalho que propôs um *framework* de desenvolvimento de aplicações utilizando OOHDM como método de desenvolvimento foi o OOHDM-Java em [Pizzol, 98], logo seguido por OOHDM-Web em [Moura, 99].

O *framework* OOHDM-Java2, apresentado em [Jacyntho, 01], definiu um *framework* MVC e uma arquitetura de implementação para o desenvolvimento de aplicações OOHDM na plataforma J2EE.

Os trabalhos de [Szundy, 2004] e [Moura, 2004] definem o desenvolvimento de uma aplicação utilizando SHDM como método de desenvolvimento. Porém, como todos os trabalhos citados acima, requerem um grande esforço de implementação e não permitem que a aplicação seja desenvolvida de forma visual.

Os trabalhos descritos acima serviram de inspiração para o desenvolvimento do HyperDE [Nunes, 2005]. O objetivo HyperDE é fornecer uma integração entre um *framework* MVC e um ambiente de desenvolvimento que permita a protipação rápida de aplicações hipermídia, baseadas nos métodos OOHDM e SHDM.

Contudo no HyperDE o desenvolvimento dos modelos OOHDM/SHDM é efetuado através de formulários e não incluem o mapeamento de modelos abstratos de interface conforme especificados em [Moura, 2004].

O HyperDE utiliza 2 tipos de controladores para dar suporte as tarefas da aplicação. O *Navigational Controller* (controlador de navegação) e um *CRUD Controller* (controlador de retaguarda). O *Navigational Controller* é responsável pelas tarefas de construção de índices, apresentação de contextos e controle da navegação da aplicação. O *CRUD Controller* é responsável pela criação e manutenção dos objetos navegacionais.

No SHDM .Net o controlador executa a função de identificar o contexto ou estrutura de acesso responsável pelo tratamento da requisição e, após o

processamento da requisição, seleciona a interface responsável pela apresentação do resultado.

O SHDM .Net teve como inspiração o HyperDE em conjunto com os trabalhos de [Moura, 2004] e [Szundy, 2004]. Buscando uma melhoria no processo de desenvolvimento através da criação dos modelos de forma diagramática, unificando também a definição da interface abstrata integrada com o restante do processo de desenvolvimento.

WebRatio [Acerbis et al., 2004] é um *framework* para engenharia de aplicações Web que suporta geração de código. É baseado na linguagem proprietária WebML [Ceri et al., 2002]. Embora seja um *framework* bastante completo a interface é produzida baseada em templates XML e os aspectos navegacionais e de apresentação não são claramente separados um do outro.

UML-based Web Engineering (UWE) [Koch et al., 2001] é uma extensão do UML usada para dar suporte ao desenvolvimento de aplicações Web. ArgoUWE é a ferramenta CASE para o desenvolvimento utilizando UWE e foi desenvolvida como uma extensão da ferramenta ArgoUML.

Hera [Vdovjak et al., 2003] é uma metodologia de desenvolvimento baseado em modelos para aplicações Web. Utilizando formatos de descrição da SemanticWeb para representar os modelos.

ArgoUWE e Hera são diferentes da nossa abordagem por utilizar uma extensão da UML e RMM [Isakowitz et al., 1997] respectivamente, como metodologias de desenvolvimento.

Nenhuma das abordagens acima permite a manipulação direta dos elementos do modelo através das DSL geradas como o SHDM. .Net permite.

A modelagem da interface no UWE possui apenas 4 elementos (texto, âncora, botão e texto) não contemplando a representação abstrata de elementos. Nem o mapeamento para elementos concretos de uma tecnologia.

WebRatio possui uma forma de desenvolver a interface bem completa. A interface é produzida utilizando templates customizáveis que podem ser desenvolvidos utilizando outras tecnologias importadas na ferramenta. Contudo a ferramenta não possui um modelo abstrato de interface que pode ser mapeado em diversas tecnologias. O diagrama onde a interface é desenvolvida e customizada é dependente de tecnologia utilizada na apresentação.

6.2.Contribuições

As principais contribuições deste trabalho são:

- Fornecer um *framework* de desenvolvimento de aplicações hipermídia, utilizando OOHDM / SHDM como método de desenvolvimento, onde o usuário possa desenvolver uma aplicação hipermídia se preocupando apenas com a modelagem, e toda a aplicação é gerada a partir dos modelos definidos pelo usuário.

- Desenvolvimento de uma ferramenta orientada a modelos que efetivamente produz a aplicação e onde a forma de construção da ferramenta também é orientada a modelos.

- Definição de uma forma diagramática para representação do modelo de classes navegacionais, modelo de contextos navegacionais, modelo de interface abstrata e modelo de interface concreta, permitindo um aumento do nível de abstração do usuário no desenvolvimento dos modelos OOHDM/SHDM, propiciando também uma redução na repetição da definição de elementos introduzida pela representação gráfica e pela interpretação do posicionamento dos objetos.

- Criação de uma forma visual de desenvolver uma interface com independência de tecnologia de geração, possibilitando que a interface seja produzida em diversas tecnologias.

- Definição de uma ontologia de *widgets* concretos permitindo que o desenvolvedor consiga, através de um mapeamento entre elementos abstratos e elementos concretos, produzir uma interface concreta a partir de um diagrama, sem precisar codificar a interface.

- Criação de uma ferramenta de modelagem de interface abstrata integrada com o restante do processo de desenvolvimento.

- Desenvolvimento um Diagrama de Interface Concreta que produza a interface baseado em uma instância de ontologia de interface concreta.

- Suporte genuíno do mapeamento de modelos abstratos de interface conforme especificados em [Moura, 2004].

6.3.Trabalhos Futuros

- Produção da aplicação independente da tecnologia, permitindo que a aplicação produzida funcione em qualquer plataforma. O esforço de implementação desta funcionalidade é moderado, já que são necessárias alterações no script que interpreta a DSL-Microsoft e no controlador responsável por capturar as requisições.

- Incluir suporte a modelagem e implementação de navegação facetada e do conceito de sub-relacionamento. O esforço de implementação desta funcionalidade é alto, pois são necessárias alterações nos editores dos modelos, definindo uma interface para a modelagem das facetas e do conceito de sub-relacionamento ocasionando uma alteração em todo o fluxo do desenvolvimento para adicionar suporte a modelagem de facetas e sub-relacionamento com o objetivo de produzir automaticamente o código fonte.

- Mapeamento de modelo conceitual/navegacional como em Szundy, 2004. O esforço de implementação desta funcionalidade é alto, pois serão necessários o desenvolvimento de um diagrama para a produção do modelo conceitual e a alteração do modelo navegacional para que seja possível fazer o mapeamento do modelo navegacional sobre o modelo conceitual.

- Inclusão de mecanismos referentes à segurança: autenticação e controle de permissões O esforço desta modificação é baixo visto que os diagramas já foram desenvolvidos contemplando a utilização de mecanismos de segurança, faltando para a implementação a inclusão deste suporte no interpretador do diagrama, produzindo assim o código fonte com suporte a estes mecanismos.

- Possibilitar a definição de regras de inferência semântica através das interfaces da aplicação. O esforço desta modificação é alto, pois será necessário adicionar suporte a regras de inferência na linguagem de definição dos contextos e estruturas de acesso bem como na parte de persistência da ferramenta.

- Inclusão do suporte a engenharia reversa, ou seja, interpretar um modelo de classes navegacionais ou a interface concreta de uma aplicação e produzir os diagramas OOHDM / SHDM, facilitando o desenvolvimento de aplicações utilizando a estrutura da codificação pré-existente.