6
Conclusão

Neste capítulo, estaremos inicialmente descrevendo, de forma resumida, alguns trabalhos relacionados ao trabalho realizado nesta dissertação. Em seguida, iremos analisar as principais contribuições feitas através da pesquisa realizada e do ambiente HyperDE. Finalizando, veremos alguns possíveis futuros trabalhos, relacionados ao ambiente e ao framework, que podem potencialmente ser de interesse dentro desta área de pesquisa.

6.1. Trabalhos Relacionados

O framework OOHDM-Java2, apresentado em [Jacyntho, 01], definiu um framework MVC e uma arquitetura de implementação para o desenvolvimento de aplicações OOHDM em plataforma J2EE. Apesar de robusto no suporte dado às primitivas do método OOHDM, o principal objetivo deste trabalho foi proporcionar a desenvolvedores da plataforma J2EE um ganho de produtividade no desenvolvimento de aplicações baseadas no método OOHDM através do reuso dos componentes em linguagem Java. Desta forma, todos os passos de modelagem e implementação de uma aplicação OOHDM devem ser executados programaticamente, geralmente através do subclasseamento de dezenas de classes, e portanto, o esforço de programação requerido neste processo ainda é bastante considerável e complexo. Este trabalho não provê um ambiente visual e ferramentas adicionais para o desenvolvimento rápido de aplicações OOHDM, tampouco oferece suporte ao uso de tecnologias da web semântica, já que este framework precede a criação do método SHDM. Contudo, este trabalho foi de grande utilidade para análise e modelagem do framework MNVC do ambiente HyperDE.

O OOHDM-Java2 teve sua própria concepção idealizada a partir de outros trabalhos, como OOHDM-Java em [Pizzol, 98] e OOHDM-Web em [Moura, 99], que também propunham a criação de um framework e um ambiente de desenvolvimento para aplicações OOHDM, respectivamente. Outros trabalhos

fortemente relacionados a estes foram apresentados em [Medeiros, 01], através da linguagem de marcação OOHDM-ML, que define um DTD XML para a declaração do modelo navegacional de uma aplicação OOHDM, e da ferramenta OOHDM-Xweb, responsável por traduzir os documentos escritos sob esta DTD para tabelas do ambiente OOHDM-Web.

Todos estes trabalhos não chegaram a fornecer, de forma unificada, uma pilha completa de camadas em conjunto com um ambiente de desenvolvimento, que possibilitasse a criação e execução de uma aplicação OOHDM, de ponta a ponta. Seus objetivos bem definidos visavam a solução específica de partes do problema, e neste ponto, foram valiosos para inspirar os conceitos utilizados na construção do ambiente HyperDE. Outro aspecto a notar é que todos eles precederam a criação do método SHDM e de tecnologias semânticas e, por isso, seus mecanismos de persistência eram geralmente baseados em bancos de dados relacionais, utilizando linguagem de consulta SQL.

O método SHDM definido por [Lima, 03] define primitivas para construção de modelos conceituais e navegacionais de aplicações hipermídia semânticas através de um modelo de dados orientado a objeto, que podem ser representados através de ontologias utilizando linguagem OWL. O modelo navegacional apresentado por [Szundy, 04] estende este modelo, através de uma representação dos dados já baseada em RDF, fornecendo formas de mapear diretamente o modelo conceitual em RDF em um modelo navegacional, em linguagem RDF(s) ou OWL. A arquitetura de implementação apresentada neste trabalho utiliza a plataforma Java e o framework Jena2 como forma de construir estes modelos navegacionais. No entanto, o framework definido por esta arquitetura não fornece uma linguagem específica de domínio para manipulação do modelo e não define o mecanismo de visualização das instâncias do modelo.

Este trabalho foi complementado através do modelo de interfaces abstratas apresentado em [Moura, 04], que define uma forma de mapear a visualização do modelo navegacional através de elementos de interface abstratos e que então podem gerar elementos de interface concretos, dependendo do dispositivo de acesso e da plataforma cliente. A combinação dos trabalhos apresentados por [Szundy, 04] e [Moura, 04] permitem o desenvolvimento de uma aplicação completa baseada no método SHDM, no entanto, de forma similar aos outros trabalhos já citados, requerem um esforço de implementação grande e não

provêem um ambiente visual para o desenvolvimento da aplicação. Contudo, ressaltamos, que a análise realizada sobre estes trabalhos serviram de forte base para construção do ambiente HyperDE.

Alguns outros trabalhos relacionados que podemos citar incluem o projeto Hera, desenvolvido pela Technische Universiteit Eindhoven, e o WebML, da Universidade Politécnica de Milão. O primeiro tem como objetivo desenvolver um método e fornecer ferramentas para construção de aplicações hipermídia para acesso a sistemas de informações (web information systems - WIS), baseados em tecnologias semânticas como RDF, RDF(s) e OWL. O WebML define um processo e uma notação gráfica, baseada em UML, especificamente destinada ao projeto de aplicações Web. Ela propõe a utilização de 3 modelos principais: modelo de dados, modelo hipertextual e modelo de apresentação.

6.2. Contribuições

O principal objetivo do 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. Atingir este objetivo significa propiciar, aos usuários desses métodos e dos modelos promovidos por eles, ferramentas capazes de torná-los mais produtivos e focados, permitindo que eles concentrem seus esforços na modelagem de suas aplicações e não em construção de infraestrutura para desenvolvimento e execução das mesmas.

Muitas vezes um método, modelo ou arquitetura só consegue atrair uma comunidade significativa quando estão disponíveis boas ferramentas para seus potenciais membros. Acreditamos que o HyperDE seja um passo significativo nessa direção, pois oferece um ambiente integrado, em que o esforço para implementação de uma aplicação OOHDM/SHDM mantém-se baixo durante todas as etapas do processo.

Outra característica interessante é que este suporte ao processo de desenvolvimento ocorre de forma iterativa e incremental, o que geralmente ajuda a reduzir o tempo no ciclo de desenvolvimento, fornecendo resultados tangíveis a cada passo, gradualmente, através de refinamentos sucessivos. Esta característica credencia o ambiente a ser utilizado em associação a processos de

desenvolvimento ágeis, como Extreme Programming [Beck & Andres, 2004], Scrum [Schwaber, 2004], Crystal Clear [Cockburn, 2004], entre outros.

O ambiente HyperDE também pode ser visto como um exemplo de "Model Driven Architecture" (MDA), onde uma aplicação é implementada primariamente através da definição de um modelo independente de plataforma ("Platform Independent Model" - PIM) e então convertido para execução em uma arquitetura de implementação específica. Sobre este aspecto, gostaríamos de esclarecer que o HyperDE não está de acordo com as especificações oficiais da MDA, segundo o OMG ("Object Management Group") e por empresas e instituições ligadas ao uso de UML e ferramentas CASE, como Rational, Sun, IBM, Microsoft, entre outras, mas apenas que seu estilo arquitetônico condiz com o que é promovido pela MDA.

Ainda sobre isso, acreditamos que o uso deste tipo de abordagem, onde tenta-se definir todos os detalhes da navegação de uma aplicação sem o uso de implementação em linguagem de programação, tal qual tenta-se fazer através de ferramentas CASE, não é aplicável à grande maioria das aplicações usuais de mercado pois impõe sérias restrições ao processo de desenvolvimento, exigindo um grau de planejamento e complexidade bem acima da média e ficando fora dos objetivos deste ambiente.

Um exemplo claro desta não-conformidade do HyperDE em relação a MDA é a possibilidade de programação através de código-fonte de operações e atributos computados de classes navegacionais e também em templates de visões customizadas, e que justamente conferem ao ambiente uma qualidade de flexibilidade e dinamicidade. No entanto, esta qualidade deve ser utilizada com responsabilidade pois tem como desvantagem o risco potencial de introduzir inconsistências não previstas no modelo.

Analisando especificamente a camada de modelo navegacional do ambiente HyperDE, podemos perceber uma qualidade única em relação a outros ambientes similares e frameworks utilizados em trabalhos relacionados. O ambiente gera dinamicamente uma linguagem específica de domínio (DSL – *Domain Specific Language)* para o modelo navegacional e que permite um estilo de programação muito mais natural e intuitivo. O benefício maior disto é a simplicidade com que podemos manipular os objetos da aplicação conhecendo praticamente apenas a DSL baseada no modelo.

Vejamos um exemplo, tomando como base o trecho de código-fonte em Java que utiliza o framework para acesso a modelo de dados semânticos (ontologias) Jena [McBride, 2001] e que não possui essa característica:

```
DAMLModel model = … // code that loads the VCARD ontology
                    // and some data based on that ontology
DAMLClass vcardClass =
    (DAMLClass) model.getDAMLValue(vcardBaseURI+"#VCARD");
DAMLProperty fnProp =
    (DAMLProperty) model.getDAMLValue(vcardBaseURI+"#FN");
DAMLProperty emailProp =
    (DAMLProperty) model.getDAMLValue(vcardBaseURI+"#EMAIL");
Iterator i = vcardClass.getInstances();
while (i.hasNext()) {
    DAMLInstance vcard = (DAMLInstance) i.next();
    Iterator i2 =
        vcard.accessProperty(emailProp).getAll(true);
    while (i2.hasNext()) {
        DAMLInstance email = (DAMLInstance) i2.next();
        if (email.getProperty(RDF.value).getString().equals(
            "amanda_cartwright@example.org" ) ) {
            DAMLDataInstance fullname =
                (DAMLDataInstance) vcard.accessProperty(fnProp).getDAMLValue();
            if ( fullname != null )
                System.out.println("Name: "+ fullname.getValue().getString());
        }
    }
}
```

Podemos reparar no exemplo acima, como é necessário o conhecimento do metamodelo DAML e apresentado na seção 2.5 da dissertação, cuja função é iterar sobre uma coleção de objetos do tipo vCard (uma classe que representa um cartão de visita e compõe uma ontologia para descrição de dados pessoais de indivíduos) e imprimir o nome da pessoa caso seu email seja de determinado valor. Este trecho poderia ser escrito em Ruby, utilizando a linguagem específica de domínio gerada dinamicamente pelo framework do HyperDE, da seguinte forma:

```
VCard.find_all.each { |vc|
    vc.emails.each { |email|
        print vc.fn if email == “amanda_cartwright@example.org” && vc.fn?
    }
}
```

No nosso framework, a classe “VCard" torna-se uma classe nativa da linguagem Ruby, dinamicamente gerada em tempo de execução pelo ambiente HyperDE, assim como as propriedades e métodos (“fn”, “fn?” e “emails”), apenas baseando-se no modelo de dados da aplicação. Os outros métodos usados (“each” e “find_all”) são nativos da linguagem Ruby e da camada de persistência utilizada pelo framework HyperDE (classes “Array” e “ActiveRecord::Base” respectivamente). A diferença de expressividade e legibilidade nos dois casos é substancial e é possível graças à dinamicidade do ambiente na criação de linguagens específicas de domínio para o modelo da aplicação e da coesão da linguagem Ruby.

O ambiente HyperDE também apresentou algumas extensões ao meta-modelo navegacional dos métodos OOHDM e SHDM. A introdução da modelagem e implementação de atributos computados e a forma de implementação de operações em classes navegacionais foram as extensões mais significativas. Além dessas, o ambiente promoveu pequenas adições de meta-propriedades em algumas meta-classes do modelo navegacional, com o objetivo de direcionar o ambiente de desenvolvimento em suas interfaces com o usuário e na geração de linguagens específicas de domínio, além de facilitar o uso de algumas funções de auxílio de interface, disponíveis na camada de visão do framework.

O ambiente HyperDE fez uso de um banco de dados RDF para a persistência de seus objetos, cujo acesso é possível através de protocolo REST, ou seja, através de canal e métodos HTTP e marcação XML. Além disso, os controladores e ações da camada de controle do ambiente também dão suporte a acesso via protocolo REST. Desta forma, é possível utilizarmos arquiteturas físicas flexíveis para execução do ambiente, onde podemos ter o servidor de banco de dados, o servidor da aplicação e a estação cliente da aplicação distribuídas em máquinas dedicadas ou associadas de formas não-convencionais, como podemos ver no diagrama a seguir:

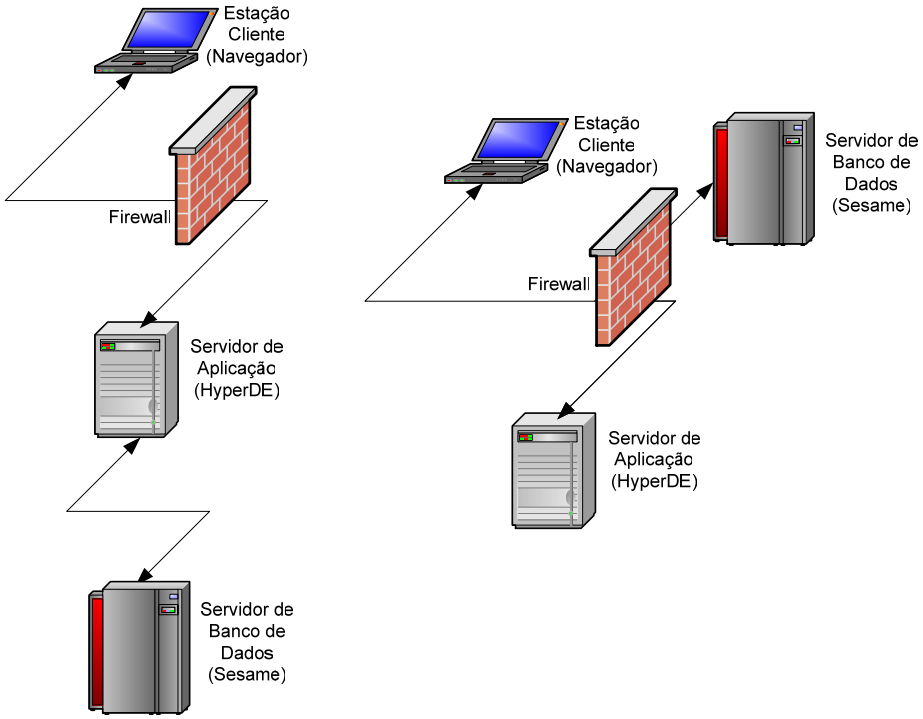

Figura 62 - Exemplos de arquiteturas físicas distribuídas possíveis para uso do ambiente

Na figura acima, no exemplo da esquerda, vemos uma configuração usual em cascata, onde o cliente da aplicação tem acesso remoto apenas ao servidor de aplicação, devidamente protegido por trás de um firewall, e este por sua vez acessa o servidor de banco de dados, possivelmente localizado na mesma sub-rede. Já na configuração da direita, vemos uma distribuição onde o cliente realiza o acesso remoto à aplicação no HyperDE, mas detém ainda a tutela sobre o banco de dados, que está localizado na mesma sub-rede que a estação cliente. Esta configuração possibilita modelos de serviços onde apenas o servidor de aplicação é contratado, sendo responsabilidade do cliente manter e assegurar o banco de dados da aplicação. Note que por usarmos apenas canal e métodos HTTP na comunicação entre os servidores e clientes, não há qualquer dificuldade em trafegar os dados através dos firewalls.

O uso pelo ambiente HyperDE de uma base de dados semântica, baseada em RDF e RDF(s), possibilita a modelagem navegacional através de ontologias, como proposto no método SHDM e confere a este esquema de dados a característica de interoperabilidade entre diferentes ontologias e sistemas de informação, pois estão armazenados na base e disponíveis para consultas e manipulação, tanto os dados da aplicação, como também os meta-dados que definem a ontologia do ambiente,

ou seja, do metamodelo OOHDM/SHDM, e os meta-dados que descrevem o modelo navegacional da própria aplicação desenvolvida. Portanto, há pouca ou nenhuma diferença na mecânica de acesso a dados e meta-dados, restando apenas a diferença semântica de cada tipo de dado, que por sua vez, é descrita pelos próprios meta-dados.

A base de dados usada também implementa regras de inferência nativas para a linguagem RDF(s), de forma a tornar possível a resoluções de consultas implícitas sobre o metamodelo da aplicação, além de fornecer os mecanismos para implementação de regras de inferência customizadas, visando estender estas capacidades, de forma específica para cada ontologia.

A camada de visão do framework HyperDE provê um modelo de implementação de visões customizadas, associadas às primitivas do ambiente, que permite a construção de templates flexíveis para visualização dos nós da aplicação. Os mecanismos oferecidos para construção destes templates, através de componentes de interface reutilizáveis e uma biblioteca de funções auxílio, permite ao usuário do ambiente descrever uma visão concreta que pode se aproximar bastante do modelo abstrato de interface, visando reduzir a necessidade ou facilitando mapeamentos entre modelos de interfaces abstratas e suas versões concretas.

Esta dissertação oferece ainda, através da implementação de uma aplicação de exemplo, uma referência completa para utilização de todos os recursos disponíveis no ambiente HyperDE de forma a facilitar e ilustrar o entendimento do mesmo de forma prática, e também como prova e validação dos conceitos promovidos pelo ambiente.

Como artefato de software, o ambiente HyperDE tornou-se, ao longo do processo de revisão desta dissertação, uma ferramenta disponível em regime de software livre (ver [HyperDE] e apêndice D), através do laboratório TecWeb da PUC-Rio, para uso na comunidade acadêmica com fins de pesquisa na área de desenvolvimento de aplicações web e web semântica, e já vem recebendo contribuições e melhorias importantes decorrentes da colaboração voluntária.

Por último, é importante citarmos algumas restrições do ambiente HyperDE. Guiado em seus princípios de projeto para manter-se dinâmico e flexível, o ambiente faz uso de introspecção constante sobre o metamodelo da aplicação de forma a refletir instantaneamente quaisquer alterações realizadas no modelo da

aplicação. Em conseqüência, a regeração freqüente dos modelos impõe restrições de seu uso em ambientes de produção, pois compromete o desempenho da aplicação. Além disso, atualmente o ambiente não oferece de forma nativa outros mecanismos necessários para uso em ambiente de produção, como mecanismos de segurança, “cache”, gerenciamento de sessões de uso, entre outros. Tais deficiências e restrições poderão ser abordadas em trabalhos futuros, como veremos a seguir.

6.3. Trabalhos Futuros

Os trabalhos futuros, relacionados ao ambiente HyperDE, podem ser classificados em dois grupos: melhorias no ambiente e no framework, sob a ótica de implementação de novas funcionalidades e melhoria das existentes, e extensões do ambiente e seu modelo de dados para incluir aspectos da modelagem dos métodos OOHDM e SHDM não implementados na versão atual.

Sobre a implementação de novas funcionalidades e melhoria de existentes, podemos destacar:

- Compilação da aplicação para um modelo estático, incluindo sua linguagem específica de domínio e visões customizadas, de forma a melhorar o desempenho da aplicação e torná-la independente do ambiente de desenvolvimento, possibilitando dessa forma, o uso da aplicação compilada em ambientes de produção.
- Inclusão de mecanismos referentes a segurança: autenticação e controle de permissões.
- Possibilitar a definição de regras de inferência semântica através das interfaces da aplicação.
- Adicionar aos controladores da aplicação o suporte ao protocolo SOAP, aderindo ao protocolo padrão de WebServices da indústria.
- Incluir adaptadores para outros bancos de dados, sejam eles relacionais, semânticos ou orientadas a objetos, além do Sesame.
- Verificação de consistência/integridade do modelo e meta-modelo.
- Tratamento de erro e exceções.

Sobre extensões referentes a modelagem e a aspectos mais globais do ambiente, podemos citar:

- Incluir o suporte ao mapeamento entre modelos conceituais e navegacionais, conforme proposto em [Szundy 04]
- Incluir suporte a modelagem e implementação de navegação facetada.
- Suportar genuinamente o mapeamento de modelos abstratos de interface conforme especificados em [Moura 04].
- Melhoria dos elementos de interface e modelagem navegacional para realizar melhor a captura de interações com usuários, através de formulários definidos no modelo navegacional da aplicação.
- Adicionar a capacidade de navegação adaptativa diretamente no modelo navegacional da aplicação.
- Suportar a modelagem de ontologias através de linguagens semânticas mais expressivas, tais quais as linguagens DAML+OIL e OWL.