3 Arquitetura de Implementação

Essa dissertação apresenta uma proposta de uma arquitetura orientada a modelos (*Model Driven Architecture* - MDA) [Brown, 2004] cujos modelos são descritos em diagramas e processados pelo ambiente. O principal objetivo é facilitar o desenvolvimento de uma aplicação hipermídia, utilizando SHDM como método de desenvolvimento, permitindo que o arquiteto da aplicação concentre seus esforços na modelagem da aplicação.

A modelagem da nova aplicação é efetuada com o auxílio de um ambiente gráfico onde os diagramas SHDM são criados. Estes diagramas também são criados utilizando uma ferramenta baseada em modelos. Com base nos diagramas criados pelo arquiteto uma aplicação hipermídia será produzida, permitindo um processo ágil e incremental do desenvolvimento de aplicações, que proporciona um freqüente *feedback* dos usuários sobre a aplicação que está sendo produzida.

Os dados criados e acessados pelas aplicações e o modelo de classes correspondente são armazenados em um banco de dados RDF/RDFS.

Será disponibilizada uma forma de modelagem e desenvolvimento da interface independente do dispositivo e tecnologia a ser utilizada como *front - end* e sem esforço de alteração e adequação da interface a tecnologia a ser utilizada na apresentação. Esta modelagem da interface também será efetuada de forma diagramática, com o auxílio de elementos de interface abstratos (*widgets* abstratos). Os elementos de interface abstratos são mapeados posteriormente para elementos de interface concretos (*widgets* concretos), responsáveis pela representação da informação, conforme especificado em [Moura, 2004]. Contudo, este mapeamento será feito baseado em uma ontologia de *widgets* concretos proposta por esta dissertação.

Para cada modelo desenvolvido (Classes Navegacionais, Contextos e Interface) é produzida uma linguagem específica de domínio (*Domain Specific Language* - DSL) [Van Deursen, 2000] com o objetivo de facilitar a programação e a modelagem. Esta DSL permite que a validação dos modelos seja efetuada de forma imediata, ou seja, no momento da modelagem.

Um processo de desenvolvimento de software é um guia de como um produto de software deve ser construído do início ao fim. Geralmente, um processo de software define atividades chaves para direção, identificação de artefatos que devem ser criados e descrição de papéis (responsabilidades) que um ou mais pessoas devem assumir em um projeto de software. A tabela abaixo representa os artefatos produzidos e consumidos em cada etapa do desenvolvimento utilizando o SHDM .Net.

Fase	Modelagem de Classes Navegacionais	Modelagem dos Contextos Navegacionais	Modelagem da Interface Abstrata	Renderização da Interface
Artefato de Entrada	-	Diagrama de Classes Navegacionais	Diagrama de Classes Navegacionais e de Contextos Navegacionais	Diagrama de Interface Concreta
Produzido (Usuário da Aplicação)	Diagrama de Classes Navegacionais	Diagrama de Contextos Navegacionais	Diagrama de Interface Abstrata	Mapeamento dos *widgets* abstratos nos concretos com auxílio da ontologia de *widgets* concretos
Artefatos de Saída (Gerado pela Ferramenta)	DSL-SHDM e programa na DSL-Microsoft do diagrama de classes navegacionais	DSL-SHDM e programa na DSL-Microsoft do diagrama de contextos navegacionais	Programa na DSL-Microsoft do diagrama de interface abstrata e diagrama de interface concreta	Interface Concreta

Tabela 1 – Fases do desenvolvimento de uma aplicação e artefatos produzidos.

3.1.MDD

Desenvolvimento dirigido por modelos (MDD) [Mellor et al. 2003] é uma abordagem para o desenvolvimento de sistemas que aumenta o poder de representação do modelo para atender um conceito específico, aumentando o poder de abstração e reduzindo o tempo de desenvolvimento. O MDD estimula o uso da abordagem dirigida para automaticamente produzir o sistema modelado.

Atualmente, uma implementação de MDD que tem atraído à atenção dos desenvolvedores é o Model-Driven Architecture (MDA) [Brown, 2004] proposta e patrocinada pelo Object Management Group OMG. O modelo de aplicações definido pelo MDA deve ser independente de plataforma e capaz de ser transformado em uma aplicação. O MDA está focado na funcionalidade e no comportamento da aplicação, mantendo os modelos independentes da tecnologia utilizada para implementá-los.

3.2.RDF/RDFS

O Resource Description *Framework* (RDF) [Manola, 2004] é um padrão para descrever, utilizando metadados, qualquer informação existente na Web, ou seja, é um padrão utilizado para descrever informações sobre informações. Logo a informação passa a ser entendida dentro do contexto no qual foi produzida permitindo que uma mesma informação esteja disponível para aplicações diferentes da qual foi projetada, proporcionando uma maior interoperabilidade entre as aplicações

RDF Schema (RDFS) [Brickley, 2004] é uma extensão semântica do RDF que proporciona mecanismos para descrever conjuntos de indivíduos relacionados e relacionamentos entre estes conjuntos. Estes conjuntos são utilizados para determinar características de outros grupos como domínio de propriedades.

3.3.Arquitetura de Desenvolvimento

O *Framework* SHDM .Net será desenvolvido utilizando a seguinte arquitetura:

- Linguagem de programação: C#
- IDE de desenvolvimento: Visual Studio 2005 Professional
- Banco de dados: SQL Server 2000

3.3.1.Ferramentas de apoio ao desenvolvimento

Em cada camada do *framework* diversas ferramentas foram utilizadas para auxiliar o processo de desenvolvimento.

3.3.1.1.Microsoft Domain Specific Language Tools (DSL-Tools)

Microsoft Domain-Specific Language Tool é um conjunto de ferramentas para criar, editar e visualizar metadados que são suportados por um *framework*,

tornando mais fácil a definição da sintaxe abstrata para um domínio de problema específico, permitindo também a construção de um editor gráfico no Visual Studio com o objetivo de auxiliar na manipulação da representação abstrata.

Linguagem Específica de Domínio (DSL) é uma linguagem de programação ou uma especificação de linguagem executável que oferece, através de abstrações e notações apropriadas, um grande poder de expressão focado, e normalmente restrito, a um problema específico.

Os metamodelos SHDM foram transformados em modelos de diagramas do Microsoft DSL Tools. Os modelos de diagramas são interpretados pelo DSL-Tools produzindo um ambiente gráfico de desenvolvimento onde as instâncias de modelos do SHDM .Net são criadas pelo desenvolvedor da aplicação. Os diagramas do SHDM .Net possuem um mapeamento semântico com os metamodelos SHDM.

3.3.1.2.SemWeb RDF library (Semantic Parser)

SemWeb é uma biblioteca de manipulação de dados no formato RDF, criada por Joshua Tauberer, que permite o acesso aos dados armazenados em memória, assim como em banco de dados como MySQL e Sqlite, dando suporte à utilização da linguagem SPARQL como consulta a base de dados RDF/RDFS [Manola, 2004],[Brickley, 2004].

SPARQL é uma linguagem de consulta a bases de dados RDF que está sendo analisada pelo World Wide Web Consortium (W3C) para se tornar o padrão de linguagem de consultas a bases de dados RDF.

SemWeb é usado como a camada de persistência do SHDM .Net

A biblioteca SemWeb foi modificada para permitir o uso do Banco de Dados SQLServer como repositório RDF.

3.4.Modelos SHDM .Net

Os modelos dos diagramas de classes navegacionais, diagrama de contextos navegacionais, diagrama de interface abstrata e do diagrama de interface concreta são criados utilizando o DSL Toolkit.

As instâncias desses modelos, que são produzidas pelo usuário da aplicação, têm uma relação muito forte com o metamodelo do SHDM, podendo

ser convertidas diretamente em modelos SHDM, através de um mapeamento simples.

Estes modelos são interpretados pelo Visual Studio .Net que produz uma ferramenta de design gráfico permitindo que o usuário desenhe graficamente os modelos SHDM .Net.

Para cada modelo feito o Microsoft DSL-Tools produz automaticamente uma linguagem específica de domínio (DSL) que permite que as instâncias modelos, criadas pelo usuário, sejam validadas entre si, reduzindo assim erros no desenvolvimento da aplicação. Para referência chamaremos a DSL gerada de forma automática pelo Microsoft DSL-Tools de "DSL-Microsoft".

A Figura 7 exemplifica o processo de criação do editor de um diagrama utilizando o Microsoft DSL-Tools.

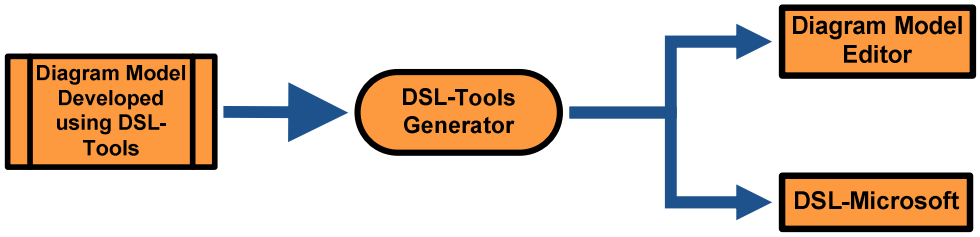

Figura 7 - Processo de construção do editor de um diagrama utilizando Microsoft DSL-Tools.

Enquanto a instância do modelo está sendo produzida, um programa na "DSL-Microsoft" está sendo criado automaticamente pelo Visual Studio. Este programa permite que um modelo utilize informações que estão disponíveis em outra instância de outro modelo. Por exemplo, quando o usuário está desenvolvendo um diagrama de contextos navegacionais, este pode referenciar objetos definidos no diagrama de classes navegacionais.

Enquanto o desenvolvedor da aplicação está criando uma instância do modelo, automaticamente uma verificação de consistência entre as instâncias de modelos é efetuada. Esta verificação utiliza o programa na DSL-Microsoft gerado de forma automática a cada instância de modelo, e tem por objetivo garantir que a instância de modelo que está sendo desenvolvida seja integra com as demais. Após a verificação de integridade entre os diagramas é produzida uma DSL da

aplicação, permitindo que o desenvolvedor escreva linhas de código enquanto está modelando. Para referência chamaremos a DSL produzida baseada nas instâncias dos modelos da aplicação de “DSL-SHDM”.

As estruturas das classes geradas a partir do modelo de classes navegacionais e suas instâncias produzidas são armazenadas em um banco de dados RDF/RDFS com o auxílio da biblioteca SemWeb.

O diagrama de interface abstrata quando processado produz um diagrama de interface concreta onde será feito o mapeamento de *widgets* abstratos para *widgets* concretos. Os *widgets* concretos são lidos de uma ontologia de *widgets* concretos, esta tem as informações sobre qual *widget* concreto pode ser utilizado por cada *widget* abstrato. Depois da realização deste mapeamento a interface concreta é produzida. O Diagrama de Interface Abstrata, Diagrama de Interface Concreta e a Ontologia de interface Concreta são contribuições desta dissertação ao método SHDM.

O processo de desenvolvimento de uma aplicação utilizando o *framework* SHDM .Net é exemplificado na Figura 8.

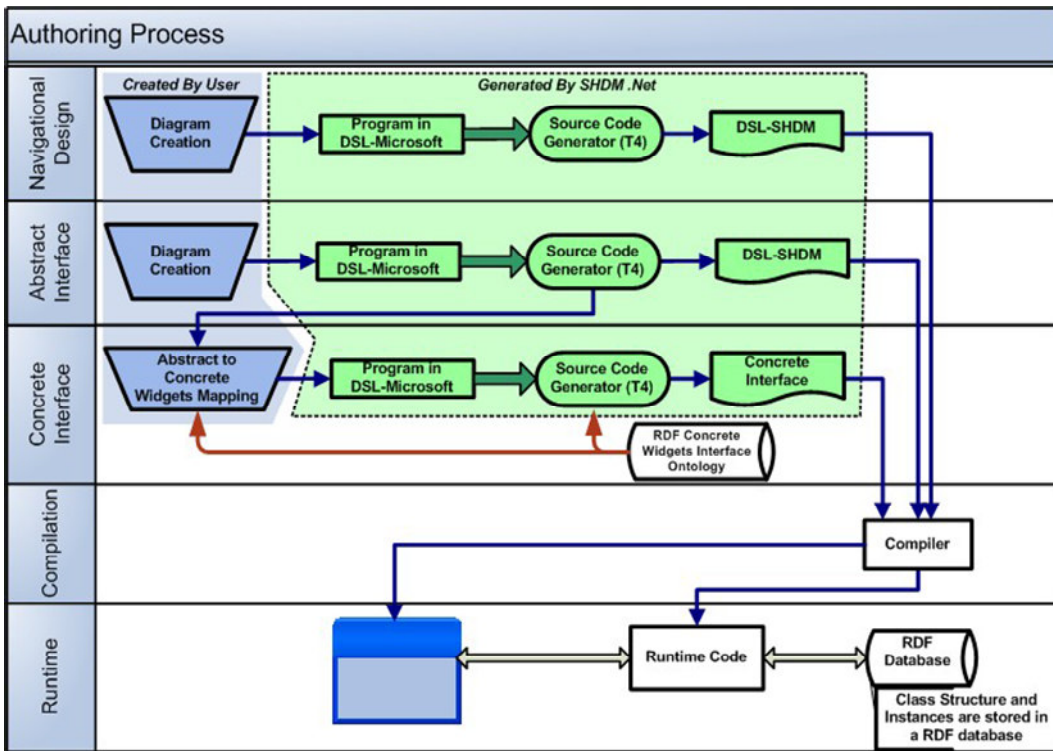

Figura 8 - Processo de desenvolvimento de uma aplicação usando o *framework*