Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Antonio Loureiro Severien

Web Application Language Engine (WALE): Geração de código para aplicações J2EE baseado em técnicas de Desenvolvimento Orientado a Modelos

Recife

2008

Antonio Loureiro Severien

(als2@cin.ufpe.br, antonio.severien@gmail.com)

Web Application Language Engine (WALE): Geração de código para aplicações J2EE baseado em técnicas de Desenvolvimento Orientado a Modelos

Trabalho de Graduação

Universidade Federal de Pernambuco

Centro de Informática

Engenharia de Software

Orientador: André Luis de Medeiros Santos

Recife

2008

LISTA DE FIGURAS

SUMÁRIO

A meu avô Paulo Loureiro, um grande homem de sabedoria sem fim e inesgotável humildade, por sempre ter acreditado em mim e por ter me ensinado que a vida deve ser curtida com pleno prazer, e que infelizmente não está aqui para comemorar comigo mais esta vitória. Eternas saudades. A luta continua companheiro!

AGRADECIMENTOS

Gostaria de agradecer primeiramente a meus pais, Paba e Mama, que lutaram junto comigo desde o inicio de minha vida até hoje para ser quem sou e ter conquistado tudo que conquistei. Agradeço também aos meus irmãos Pedro e Chico por serem os irmãos que são e que sempre estiveram junto quando precisei. Agradeço minha Tia Clausky por ter me ajudado e apoiado incondicionalmente para a realização e conclusão deste trabalho. Um agradecimento especial vai para a turma lá de casa, Meury, Maryleyne e Janis, que me trataram feito rei e competiam para ver quem era minha queridinha fazendo um sanduíche melhor que o outro enquanto trabalhava neste projeto. Agradeço a turma de 2008.1 pelas amizades criadas e pelos diversos momentos de diversão e descontração vividos juntos.

“Um passo a frente e você não estará mais no mesmo lugar.”

Chico Science

“Emancipate yourself from mental slavery.

None but our selves can free our minds.”

Robert Marley

RESUMO

O desenvolvimento orientado a modelos (Model Driven Development - MDD) se apresenta como uma solução que encurta a distancia entre as fases de análise, projeto e codificação, e agiliza o processo de implementação, modificação e manutenção do software, permitindo que analistas, programadores e testadores interajam em um nível mais alto de abstração da aplicação. A elaboração de uma ferramenta de MDD põe em prática as técnicas de MDD para o desenvolvimento de aplicações. O Web Aplication Language Enfine (WALE) faz uso de modelagem textual através de uma linguagem especifica de domínio (Domain Specific Language - DSL) que serve como um modelo para geração de código. Essa metodologia apresenta um grande potencial para o mercado que demanda cada vez mais a entrega de produtos mais rápidos, com mais qualidade e com menos erros.
Palavras chave: Model Driven Development, Model Driven Arquitecture, Domain-Specific Language, geração de código, engenharia de software.

ABSTRACT

The Model Driven Development (MDD) is presented as a solution that shortens the distance between the phases of analysis, design and coding, and speeds up the process of implementation, modification and maintenance of software, allowing analysts, programmers and testers to interact on a higher level of abstraction of the application. The development of a tool for MDD puts into practice the techniques of MDD for the development of applications. The Web Application Language Engine (WALE) makes use of textual modeling language through a Domain Specific Language (DSL) which serves as a model for code generation. This approach has great potential for the market that increasingly demands the delivery of products faster, with more quality and with fewer errors.

Key works: Model Driven Development, Model Driven Architecture, Domain Specific Language, code generation, software engeneering.

1. INTRODUÇÃO

Este trabalho tem por objetivo apresentar um novo paradigma de desenvolvimento de software que vem sendo aplicado em diversas áreas e em diversas empresas. Serão abordados os conceitos e a prática da técnica de Model Driven Development (MDD) – uma técnica baseada em modelos que servem de fonte primária para a geração de diversos artefatos de um sistema, como o código, a interface gráfica, a documentação, entre outros. Para aplicar estes conceitos é necessário o recurso a ferramentas que permitam desenvolver tal técnica. Assim sendo, serão apresentadas duas das ferramentas de MDD – programas que permitem criar outros programas específicos para o domínio de problema. Desta forma, é possível desenvolver ferramentas especializadas em áreas de domínio diferentes como aeronáutica, telecomunicação, saúde, entre diversas outras. Ferramentas de MDD permitem esta facilidade ao aumentar o nível de abstração das linguagens de programação tradicional, focando no problema do domínio em que o software será usado. Enfim, é possível a partir de um modelo, gerar uma aplicação inteira e funcional. Visando este ambiente, a empresa Estalo Consultoria e Automação decidiu criar uma ferramenta que permitisse a migração de seu software legado, de uma tecnologia antiga, para uma mais nova, além de ter uma ferramenta para criar novos sistemas com um ganho de produtividade considerável. Essa ferramenta é o Web Aplication Language Engine (WALE), especializada em gerar aplicações para o ambiente web na plataforma J2EE, com suporte a diversos frameworks como Hibernate, Java Server Faces, Seam, entre outros.

O ambiente atual de desenvolvimento de software é marcado por diferentes processos, cada um requerendo uma grande quantidade de documentação, muitas vezes não utilizadas na prática. Na implementação de soluções de software muitas das decisões ficam nas mãos dos programadores, que necessitam, com freqüência, de rapidez e não recorrem à documentação ou ao projetista ou analista, dando margem a erros ou à implementação de soluções diferentes da descrita pelo cliente.

Varias técnicas foram desenvolvidas e estão em desenvolvimento para amenizar tal problema, como processos de qualidade para reforçar o uso da documentação, programação

XP, para agilizar o processo de desenvolvimento, o uso de ferramentas CASE, para modelar de forma mais fiel a solução, retirando a responsabilidade do programador. Cada uma dessas técnicas tem seus pontos positivos e negativos.

Desenvolvimento Orientado a Modelos, ou Model Driven Development (MDD), não se apresenta como uma solução milagrosa para todos os problemas em um ambiente de desenvolvimento, mas como uma técnica de auxílio à produção, que tenta encurtar a distância entre todos os componentes presentes no ciclo de desenvolvimento de software. Esta característica permite uma integração maior entre projetistas, analistas e desenvolvedores, diminuindo o esforço de cada um em suas tarefas. MDD possibilita, a partir de uma modelagem de alto nível, a geração de grande parte dos artefatos necessários para o desenvolvimento de um software, deixando somente o que realmente é necessário para ser implementado aproveitando a criatividade e não os “braços” dos desenvolvedores.

Este trabalho está dividido em duas partes. Na primeira é feita uma revisão teórica dos conceitos envolvidos no desenvolvimento orientado a modelos e depois é apresentado um estudo de caso entre duas ferramentas usadas para criar ferramentas de desenvolvimento orientado a modelos. Na segunda parte é apresentada a ferramenta WALE com base na sua idealização, construção e aplicação prática no ambiente de produção de software.

2. REVISÃO TEÓRICA

Ao longo das últimas cinco décadas, pesquisadores de software e desenvolvedores vêm criando abstrações que ajudam a programar com base em design ao invés de lidar direto com os componentes computacionais como CPU, memória e dispositivos de rede. Estas abstrações incluem a criação de linguagem e de plataformas. Linguagens como Assembly e Fortran protegiam os desenvolvedores das complexidades de programar direto no hardware. Apesar de elevar o nível de abstração, estas linguagens e plataformas eram focadas no domínio de solução computacional, provendo abstrações para o ambiente tecnológico, ao contrário de criar abstrações para o domínio do problema que expressasse design em termos de conceitos de domínios de aplicação como telecomunicação, aeroespacial, biologia, saúde entre outros(SCHMIDT, 2006).

Grande esforço foi feito na comunidade para criar ferramentas que auxiliassem na questão de elaborar design de software em mais alto nível. Nos anos 1980, surgiram os processos e ferramentas CASE (Computer-Aided Software Engineering) possibilitando a elaboração de design gráficos de sistemas como diagramas estruturais, diagramas de estado e diagramas de fluxo de dados. CASE atraiu bastante atenção na área de pesquisa, pois prometia ser um grande auxilio no processo de desenvolvimento de software, no entanto, na prática, não foi largamente adotada e nem utilizada com todos os recursos. Devido à dificuldade de desenvolver grandes sistemas em diferentes áreas de domínio de software, além da quantidade e a complexidade do código gerado, as ferramentas CASE ficaram relegadas a serem usadas como ferramentas de modelagem gráfica com o propósito de gerar documentação para servir de guia a programadores, que nem sempre reproduziam softwares sincronizados com o projeto. Com o advento de linguagens de programação de terceira geração, como Java, C++ e C#, alguns dos problemas de integração com CASE foram amenizados.

Apesar dos grandes benefícios proporcionados pelas linguagens de programação mais modernas, ainda há muitas restrições no que diz respeito ao desenvolvimento e manutenção de grandes aplicações. O uso de plataformas de desenvolvimento trouxe várias vantagens,

no entanto, a constante evolução e o aumento da complexidade dessas plataformas requer grande esforço e tempo dos desenvolvedores para manter as aplicações em dia com as novas versões, além de necessitar um fino trato para manter a qualidade, a configuração e o esquema de implantação em dia.

Uma abordagem promissora para lidar com todo este ambiente heterogêneo e complexo, e para lidar com inabilidade de linguagens modernas de expressar conceitos de domínio eficientemente, é a criação de tecnologias de desenvolvimento orientado a modelos – MDD (SCHMIDT, 2006).

2.1 Model Driven Development (MDD)

Model Driven Development é um paradigma de desenvolvimento emergente que se propõe a solucionar vários destes problemas supracitados com a composição e integração de sistemas de grande escala. MDD eleva o desenvolvimento de software para um nível mais alto de abstração do que se é possível com as linguagens de terceira geração.

MDD usa modelos para representar os elementos de um sistema e seus relacionamentos. Os modelos servem de entrada e saída durante todo o processo de desenvolvimento até que o sistema final seja gerado (BALASUBRAMANIAN et al., 2006)

O uso de MDD para o desenvolvimento de software apresenta grandes vantagens para a empresa, para a equipe e para o ciclo de produção de software. Pensar no software através de seu domínio de atuação volta o foco do problema para o contexto do domínio em que o cliente trabalha, deixando de lado os problemas mais técnicos que sempre estão presentes no meio tradicional de desenvolvimento. Ao criar um modelo adequado e que represente um domínio especifico gera um artefato que pode ser compartilhado por todos os stakeholders do projeto. Este modelo é essencial e indispensável, pois é a partir dele que serão gerados todos os artefatos necessários para o funcionamento da aplicação. A principal função do MDD é o ganho na produtividade e qualidade no desenvolvimento de sistemas

através da transição de paradigmas de manufatura para processos industriais de desenvolvimento de software (LANGLOIS; EXERTIER; BONNET, 2006).

Uma característica importante do modelo é a de que ele possa ser lido por uma máquina, de forma que seu conteúdo possa ser acessado de maneira automatizada. A leitura e interpretação do modelo são imprescindíveis para que os artefatos necessários possam ser gerados a partir dele (YUSUF; CHESSELL; GARDNER, 2006). Em MDD os modelos não são usados apenas como diagramas e matrizes do projeto, mas como artefatos em que implementações eficientes de software usando padrões são geradas a partir de transformações de modelos em código ou outros artefatos. MDD permite uma automação que vai além da geração de código. A partir de um único modelo é possível gerar uma infinidade de artefatos, como os mostrados a seguir.

- Documentação:
  Um dos grandes problemas em um ambiente tradicional de desenvolvimento é documentação. Mantê-la em dia com o projeto é uma tarefa que demanda tempo e recursos que poderiam estar desenvolvendo outras tarefas para a evolução do software. Ao adotar MDD é possível sincronizar a documentação com o projeto, pois ela pode ser gerada a partir do modelo.

- Artefatos de teste:
  É possível gerar testes unitários básicos a partir dos modelos com suporte a frameworks reconhecidos como JUnit ou outros, dependendo de qual ferramenta de testes é adotada pela empresa.

- Scripts de build e implantação:
  Scripts de build e de implantação também podem ser gerados, tirando mais um trabalho das mãos dos desenvolvedores.

- Outros modelos:
  No ciclo de desenvolvimento existem varias etapas como análise, projeto e

implementação, e para cada etapa usa-se um tipo de modelo diferente; também há modelos que representam as diferentes partes de um sistema, interface de usuário, lógica de negócio, administração do sistema, e também há modelos que representam diferentes tarefas como testes e implantação. Dependendo do modelo é possível gerar "sub-modelos", ou algum deles, a partir de parte do modelo original.

- Aplicação de padrões:
  Padrões são soluções comuns para problemas recorrentes na industria de software. Com MDD é possível guiar a geração para que certos padrões sejam seguidos, resultando num produto final que está de acordo com o padrão A ou B.

MDD tem um grande apelo para as empresas de desenvolvimento de software, pois pode melhorar consideravelmente o processo de produção de software. Alguns dos benefícios são:

- Aumento de produtividade: Há uma redução no custo do desenvolvimento dado que grande parte do código e dos artefatos é gerada.
- Manutenção: Mudanças de tecnologias podem ser rapidamente resolvidas, pois a modelagem do sistema é feita independente da plataforma final.
- Reuso: MDD permite o reuso de padrões, arquitetura e componentes devido à geração de código estruturada sempre da mesma forma e do uso de componentes necessários para a aplicação.
- Adaptabilidade: Mudanças em funcionalidades e regras de negócio podem ser adaptadas mais facilmente, pois não é necessário mexer no sistema inteiro, somente nos pontos onde se aplicam as regras, deixando as alterações mais técnicas por conta das transformações de modelos.
- Consistência: Executar alterações manualmente é uma prática que pode levar a introdução de erros no sistema. Com MDD é possível diminuir a quantidade de erros e tornar o produto final mais consistente.
- Melhoria na comunicação entre stakeholders: Com o uso de modelos de alto nível que retratem melhor o domínio, e omitam os detalhes técnicos, é possível criar uma comunicação mais nivelada entre os stakeholders do projeto.

- Retenção de expertise: Empresas dependem de pessoas que são especialistas em determinadas áreas, e quando uma pessoa dessas sai de um projeto a empresa perde todo seu conhecimento. MDD possibilita que este conhecimento seja perpetuado em padrões e transformações, reduzindo a dependência de um integrante.
- Retardo na decisão sobre tecnologia: Muito do esforço na criação do projeto está presente na modelagem e na arquitetura do sistema, portanto decisões técnicas de qual tecnologia e plataforma serão usadas podem ser feitas mais adiante no processo.

2.2 Model Driven Architecture (MDA)

Model Driven Architecture é uma iniciativa do Object Management Group (OMG) com o intuito de formalizar os conceitos de MDD em um padrão para ser adotado pela comunidade e indústria de desenvolvimento de software. A OMG é um consorcio aberto que visa criar padrões para a interoperabilidade no espaço de aplicações empresariais. A OMG, além da iniciativa MDA, é responsável pelo Unified Modeling Language (UML) que é uma das características centrais do MDD e é largamente utilizado na modelagem de sistemas (GARDNER; YUSUF, 2006). O padrão MDA separa a lógica de negócio e aplicação da plataforma tecnológica. Modelos independentes de plataforma de uma aplicação, criados usando UML e outros padrões de modelagem da OMG, podem ser realizados através de MDA para praticamente qualquer plataforma, aberta ou proprietária, incluindo Web Services, .NET, CORBA, J2EE e outros. (OMG, 2008)

Os termos MDA e MDD às vezes são usados para designar a mesma coisa, contudo deve-se atrelar MDA à definição formal da OMG. O guia MDA da OMG descreve MDA como tendo três principais objetivos: portabilidade, interoperabilidade e reusabilidade. O intuito é atingir estes objetivos separando a especificação operacional de um sistema dos detalhes de implementação de uma plataforma especifica. Para tal o MDA define três tipos de modelos diferentes (GARDNER; YUSUF, 2006).

Computation-independet Model (CIM)

O modelo independente de computação é criado pelos analistas de negócio e representa o modelo de negócio do sistema. Este modelo descreve o objetivo e o propósito sistema, separando a lógica fundamental do sistema dos detalhes de implementação. Modelos CIM podem ser expressos em linguagens especificas de negócio ou domínio (WHITFIELD, 2007). O CIM é um modelo usado como terreno comum entre especialistas no domínio, que não tem uma visão tecnológica, e especialistas em design para construção de artefatos que satisfaçam os requisitos do domínio (OMG, 2003).

Platform-independent Model (PIM)

O modelo independente de plataforma foca na operação e estrutura do sistema, sem levar em conta detalhes da plataforma em que será implementado. É possível mostrar com o PIM uma especificação do sistema que não é alterada de uma plataforma para outra. Para modelar o PIM podem-se usar diferentes linguagens de modelagem, como UML ou outra linguagem especifica da área de atuação do sistema (OMG, 2003).

Platform-specific Model (PSM)

O modelo especifico de plataforma é uma implementação do PIM para uma ou mais plataformas. Este modelo abrange todos os detalhes técnicos e também define como o CIM será implementado em uma dada plataforma.

A transformação de modelos é o cerne do MDA, onde um modelo PIM é transformado em um modelo PSM. MDA não considera está transformação como um processo fixo de duas etapas, permitindo que um PIM possa ser um PSM referente a um modelo mais abstrato. Existe uma discussão na comunidade MDA se o PIM pode ser transformado direto para código, onde o código seria o PSM, ou se deve ser transformado para um modelo não codificado para posteriormente ser gerado o código. A Figura 1 contextualiza este processo de transformação.

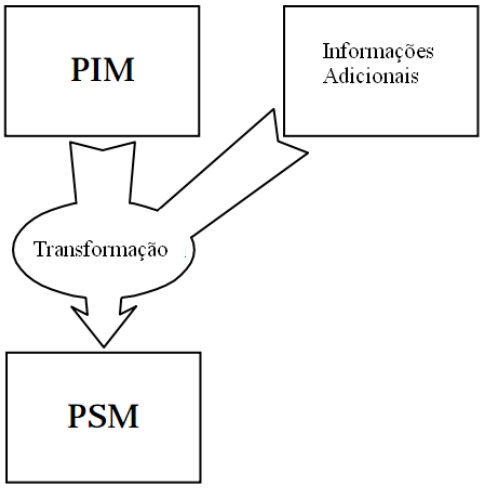

Figura 1. Transformação de PIM para PSM

2.3 Domínio

O domínio é um fator chave em MDD, pois é a partir dele que os modelos serão desenvolvidos. Existe uma técnica de design de software chamada Domain Driven Development (DDD), cujo principal objetivo é investir todas as forças no perfeito entendimento do domínio à que o software irá trabalhar. Para tal é necessário criar uma comunicação entre os especialistas do domínio e os engenheiros de software, para que o resultado final da modelagem do sistema reflita exatamente o domínio da aplicação. Quase sempre é muito difícil absorver exatamente o que o cliente quer ou necessita, por que os analistas e engenheiros têm um conhecimento muito técnico sobre como o software será desenvolvido e sempre tenta aplicar este conhecimento no entendimento do domínio. Já o cliente ou especialista no domínio não tem noção desta parte técnica, mas tem uma noção profunda de como funciona seu negócio, e para se comunicar faz uso de jargões e palavreados específicos de sua área. Como se pode ver, a comunicação é bastante difícil entre estes dois mundos, portanto necessitando de um terreno comum de comunicação. Este terreno é construído na modelagem do domínio. A modelagem do domínio é uma etapa crucial no processo de produção de software, e deve ser feita em conjunto com os especialistas de domínio através de reuniões até que se esgotem as dúvidas e o modelo final seja uma representação fiel do domínio.

A importância deste modelo não é apenas para se ter um documento de layout da aplicação, mas sim um modelo onde seja possível trabalhar o desenvolvimento do sistema. Este é o grande diferencial de MDD para outras técnicas, pois o modelo não é descartado durante o processo e se torna a matriz para a geração do código e de outros artefatos. Para tornar a geração de uma aplicação funcional, a partir de um modelo de domínio possível, é necessário o uso de linguagens especifica de domínio, geradores de código específicos de domínio e frameworks específicos de domínio. Com estes três componentes em mãos basta o desenvolvedor criar o modelo com a DSL, gerar o código e rodar a aplicação em cima do framework (STEVEN KELLY, 2008).

Definir o domínio não é uma tarefa trivial por que existem centenas e milhares de domínios diferentes, onde cada um necessita de tecnologias e requisitos específicos. Existem domínios mais abrangentes e domínios mais restritos. É possível classificar os domínios em dois tipos: horizontais e verticais. Os domínios horizontais são mais técnicos, como persistência, comunicação, interface de usuário. Já os domínios verticais são mais do domínio de negócio, como telecomunicações, seguro, banco, saúde, web, entre outros.

2.4 Domain Specific Language (DSL)

Linguagem especifica de domínio são linguagens de computação limitadas a certos domínios de problemas. As DSLs podem variar em diversas formas, tanto na sintaxe quanto na semântica, e também na forma com que elas são criadas que podem ser por meios visuais ou textuais (FOWLER, 2005). A criação de uma DSL para se trabalhar um domínio é um recurso bom e eficiente por que eleva o nível de abstração da linguagem para um dado problema. Ao focar na estrutura do domínio, e deixar de lado a “sujeira” presente no desenvolvimento da solução, a DSL se torna uma representação mais legível do que se trata o domino. Quando o domínio é mapeado em linguagens de programação comum o código fica cheio de detalhes que não são relevantes para o entendimento do domínio do problema, no entanto são imprescindíveis para o desenvolvimento da solução.

Existem diversas DSLs existentes no mundo tecnológico, e que são usadas diariamente por desenvolvedores. Muitas vezes sem nem atentar que estão usando uma DSL. Exemplos são as linguagens de markup como HTML, XML, ou as linguagens de banco de dados como SQL. O nível de abstração das DSLs depende do grau de liberdade que elas proporcionam. Existem dois tipos de DSL, a externa e a interna. A DSL externa não tem nenhuma relação com linguagens de programação e necessitam de um parser para interpretá-las, além de serem especificas com pouca liberdade. Já a DSL interna tem uma relação maior com as linguagens de programação e podem apresentar uma semântica e sintaxe similar. Esta característica permite uma maior liberdade, no entanto não é possível desenvolver qualquer aplicação com ela.

Outra característica das DSLs é a forma com que elas são desenvolvidas. É possível definir uma DSL de forma textual ou de forma visual através de componentes gráficos. A técnica adotada depende muito de que ferramentas serão utilizadas, do propósito da DSL e de quem irá utilizá-la. Se a DSL é destinada para pessoas de conhecimento técnico, como desenvolvedores, é mais provável que se utilize DSL textuais devido ao costume com o desenvolvimento em linha de código. No caso da DSL ser usada por um gerente de uma empresa, ou por um produtor de TV, é melhor recorrer aos recursos gráficos que facilitam a interação do usuário.

3 ESTUDO DE CASO

A aplicação prática do conceito de desenvolvimento orientado a modelos requer ferramentas altamente especializadas que permitam a criação de projetos totalmente baseado em modelos. MDD é uma área de pesquisa recente e ainda não foram criados padrões de ferramentas, linguagens, modelos ou processos que consolidem a técnica. No entanto, há varias empresas, grupos e comunidades investindo na criação de padrões e de ferramentas poderosas para trabalhar melhor com MDD. Uma dessas empresas é a Microsoft, que está investindo em tecnologias MDD e que desenvolveu uma ferramenta para criação de DSLs, com suporte a geração de código. Outras ferramentas estão sendo criadas por comunidades, visando ter um framework aberto para ser melhorado por qualquer um que queria contribuir com seus conhecimentos. Duas destas ferramentas são: o Visual Studio DSL Tools e o openArchitectureWare. Estas serão aplicadas, a seguir, em exemplos de implementação de uma ferramenta para TV interativa para ilustrar as diferentes características de cada ferramenta para apresentar soluções em MDD. Um sistema de TV interativa, neste exemplo, é construído com três tipos de conteúdo: menu, votos e caixa de texto. Para configurar uma TV interativa é necessário um arquivo no formato XML com o conteúdo da aplicação que será enviado por streaming digital para a TV. O propósito destes exemplos é criar uma ferramenta para produtores que permita a criação de conteúdo sem a necessidade de escrever um arquivo XML.

3.1 Visual Studio DSL Tools

O Visual Studio DSL Tools é uma extensão do Microsoft Visual Studio que permite a criação de ferramentas especificas para o domínio de interesse. Assim é possível gerar código a partir de DSL definidas neste ambiente. A ferramenta DSL Tools oferece um ambiente gráfico para definição e visualização de DSLs. Uma característica marcante dessa ferramenta é a de não estar atrelada a nenhum domínio especifico, permitindo que o designer modele DSLs para os mais diversos domínios. É possível, também, através do suporte de templates, gerar qualquer tipo de documento, desde códigos em C#, XML, a

documentos de texto. O fluxo de trabalho no DSL Tools se inicia pela definição de um modelo de domínio, onde são mapeados elementos específicos de um domínio, em seguida é criado um conjunto de elementos visuais que tem um relacionamento direto com os artefatos do domínio. A partir destes elementos é possível adicionar metadados que auxiliam na definição do comportamento e aspectos da ferramenta criada. Resta então a definição dos templates de geração que serão criados a partir dos artefatos necessários para o sistema.

Para elaborar este estudo do DSL Tools foi elaborado um exemplo de ferramenta para criação de aplicações para uma TV interativa. Este exemplo foi retirado do artigo de Steve Cook (COOK, 2007).

3.1.1 Criando um novo projeto

No Visual Studio deve-se criar um novo projeto com os componentes necessários para construir uma DSL. No menu “New Project” seleciona-se o Domain Specific Language Designer template. O designer deve seguir o wizard e escolher uma das opções pré-definidas de linguagem; neste caso a opção é Minimal Language. Após essa primeira etapa um projeto com algumas configurações iniciais é criado, no entanto não satisfaz os requisitos da TV interativa. Para elaborar a DSL é usado o DSL Designer que é um editor gráfico composto por uma estrutura em árvore de classes de domínio, que são representadas por retângulos, e de relacionamento de domínio, representado por linhas. A Figura 2 mostra o core do modelo de domínio.

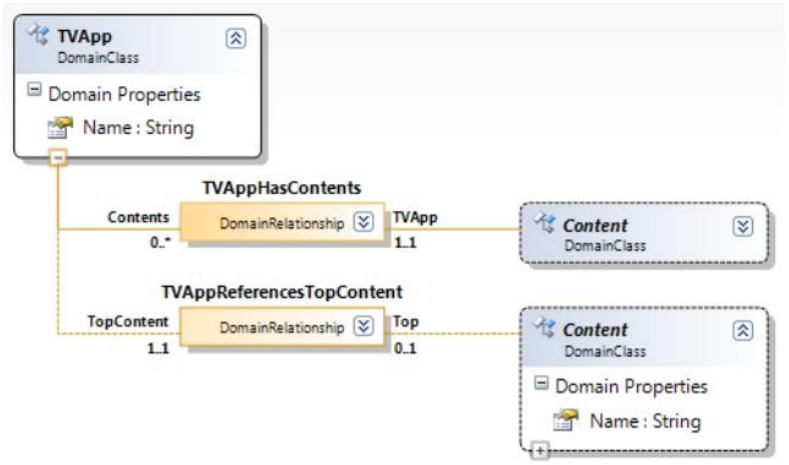

Figura 2 Core do modelo de TV interativa

Esta modelagem do domínio representa a gramática da DSL: é nela que se encontra a definição de todos os elementos da DSL, assim como a forma como se relacionam. É possível notar que a classe de domínio Content é diferente da TVApp, pois ela é uma classe abstrata. No DSL Tools as classes abstratas, como em linguagens orientadas a objetos, não podem ser instanciadas, apenas quando implementadas através de herança. As classes abstratas são caracterizadas pelo texto em itálico e a borda tracejada. A classe Content é implementada como subclasses do tipo Menu, Vote e Text, como se pode ver na Figura 3.

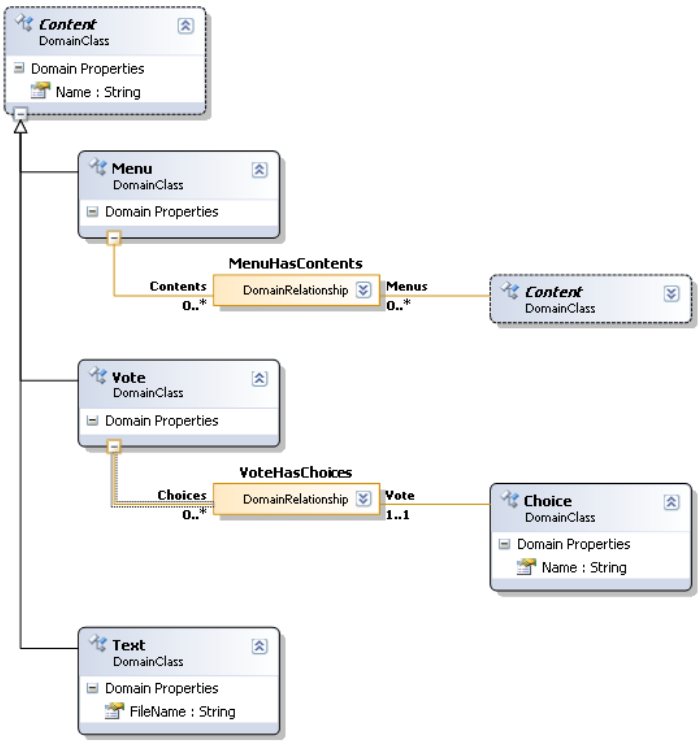

Figura 3 Subclasse de Content

A classe Menu faz uma referência a uma classe Content, o que torna está estrutura de arvore em um grafo. As relações entre classes podem ser feitas de duas formas, por uma relação de referencia ou por uma relação embutida.

Após a definição do modelo é necessário ligar cada classe do modelo a uma forma que representa uma instancia da classe na ferramenta a ser criada. Existem diversas formas no DSL Tools, no entanto neste exemplo serão usadas as formas GeometryShape para os menus e paginas de texto, e CompartmentShape para os votos. A Figura 4 mostra a definição destas formas.

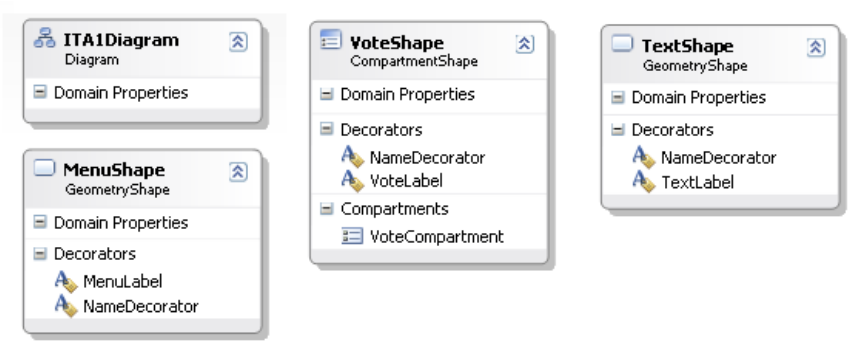

Figura 4 Definição das formas no DSL Tools.

A forma GeometryShape tem um layout geométrico que pode ser retangular, circular ou elíptico. A forma CompartmentShape contém um cabeçalho e um compartimento expansível com textos derivados do modelo. A ferramenta Diagram Map Element é usada para o mapeamento das formas com as classes do modelo. O mapeamento é representado por linhas, que podem ser exibidas ou não.

Para encerrar a primeira etapa de criação da DSL, é necessário definir a caixa de ferramentas que será usada pelo usuário final. Esta caixa de ferramenta é uma aba que contem todas as formas definidas, na qual o usuário poderá criar a aplicação de TV interativa. No DSL Explorer, dentro do DSL Designer, existe um nó Editor, acima do nó Toolbox Tabs. No Toolbox Tabs, é possível criar mais de uma seção e, em cada uma, é possível definir ferramentas que representam uma classe de domínio ou uma relação. Para a aplicação de TV interativa serão criadas quatro ferramentas: Menu, Vote, Text e MenuHasContents. A Figura 5 mostra uma aplicação de TV interativa desenvolvida com este processo.

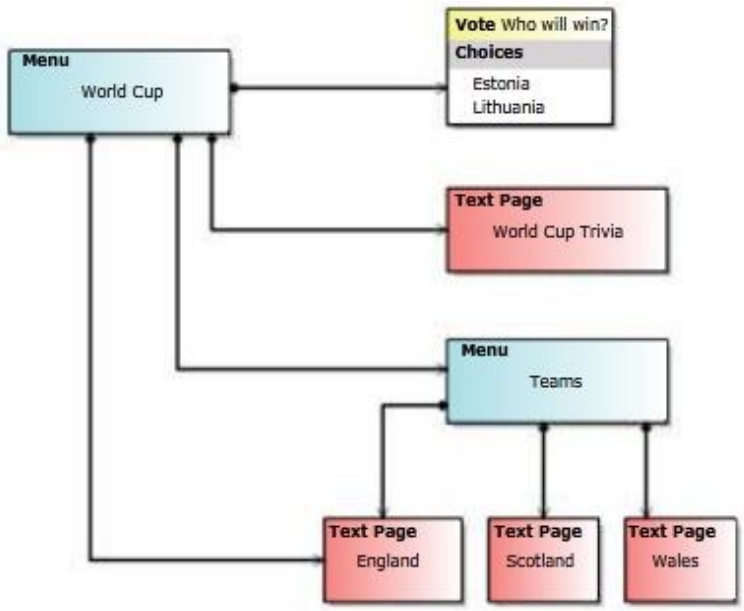

Figura 5 Exemplo de aplicação de TV interativa usando DSL Tools

É aconselhável acrescentar algumas restrições a DSL, já que o relacionamento TvAppReferencesTopContent tem uma multiplicidade mínima de 1. Este tipo de restrição é chamado multiplicity underflow checking e pode ser aplicada ao ativar a validação da aplicação. No DSL Explorer o nó chamado Validation tem as opções “Uses Load”, “Uses Menu”, “Uses Open” e “Uses Save”. Ao mudar todas as opções para true**,** o modelo de aplicação de TV interativa será validado toda vez que for aberto, carregado ou salvo.

A parte final é a mais importante, pois é a motivação pela qual a DSL foi criada. A geração de código é o produto final da ferramenta, onde os artefatos necessários para a aplicação de TV interativa serão gerados a partir dos modelos criados com a DSL. O código gerado para esta aplicação é um XML que será enviado via streaming para uma TV. No DSL Tools a geração de código é feita pela ferramenta Text Templating engine. As duas figuras a seguir mostram um exemplo de template.

```
<?xml version="1.0" encoding="utf-8"?>
<TVApp name="<#= this.TVApp.Name #>">
<#
if (this.TVApp.TopContent != null) {
  GenContent(this.TVApp.TopContent);
}
#>
</TVApp>
```

Figura 6 Código do templade em C# que chama o método GenContent()

```
<#+
public void GenMenu(Menu m) {
  this.PushIndent("  ");
#>
<Menu name="<#= m.Name #>">
<#+
foreach (Content element in m.Contents) {
  GenContent(element);
}
#></Menu>
<#+
  this.PopIndent();
}
#>
```

Figura 7 Template que chama o metodo GenMenu

Tudo que está entre as chaves <#= #>, <# #> e <#+ #> é código C# que será avaliado no contexto de execução do template. O XML final deste exemplo de aplicação é mostrado na Figura 8.

```
<?xml version="1.0" encoding="utf-8"?>
<TVApp name="World Cup 2010">
      <Menu name="World Cup">
            <Vote name="Who will win?">
                  <Choice name="Estonia">
                  <Choice name="Lithuania">
            </Vote>
            <Text Name="World Cup trivia">
                  Trivia text goes here...
            </Text>
            <Menu name="Teams">
                  <Text Name="England">England</Text>
                  <Text Name="Scotland">Scotland</Text>
                  <Text Name="Wales">Wales</Text>
            </Menu>
      </Menu>
</TVApp>
```

Figura 8 XML gerado

3.2 openArchitectureWare

O openArchitectureWare é um framework de geração para MDD/MDA implementado em Java. Ele suporta a leitura de modelos arbitrários e de uma família de linguagens para checar e transformar modelos, assim como gerar código. Os editores suportados são baseados na plataforma do Eclipse. O oAW tem um forte suporte a modelos baseados em EMF (Eclipse Modeling Framework) mas também trabalha com outros tipos de modelos, como por exemplo UML, XML e JavaBeans. No core existe uma engine de workflow permitindo a definição de workflows de geração e transformação. Inúmeros componentes de workflows preconcebidos podem ser usados para leitura e instanciação de modelos, checando-os por violação de restrições, e transformando-os em outros modelos, e finalmente para geração de código (OPENARCHITECTUREWARE, 2008).

A seguir segue um exemplo de desenvolvimento de uma ferramenta para uma TV interativa usando o openArchitectureWare. Este exemplo ilustra as características do oAW focando na definição de um metamodelo que descreve o conteúdo interativo da aplicação, a construção de um editor textual para o modelo correspondente, a validação do modelo e a geração de código XML a partir do modelo (HAASE et al., 2007).

3.2.1 Definição de um metamodelo

O primeiro passo é a definição de um metamodelo, que pode ser feito usando UML para criar um modelo visual da aplicação, como se pode ver na Figura 9. Este diagrama em seguida pode ser transformado para um arquivo Ecore utilizando o gerador uml2ecore presente no oAW.

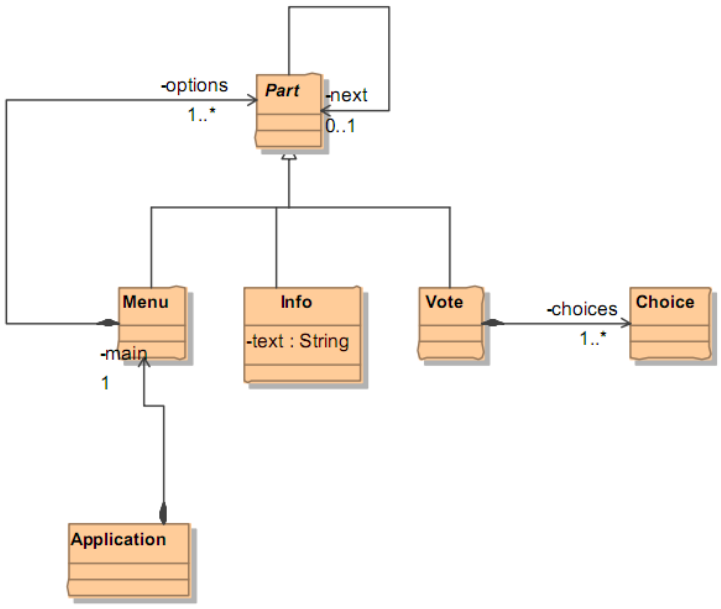

Figura 9 Modelagem em UML da gramática

Apesar do oAW fornecer este tipo de modelagem será usada uma outra importante ferramenta do oAW que é o Xtext. O Xtext é um framework de geração de uma linguagem textual de modelagem a partir de uma gramática simples expressa no formato EBNF. A seguir está a gramática que representa o modelo da aplicação.

```
ApplicationAST : name=STRING
      mainMenu = MenuAST;

PartAST : MenuAST | InfoAST | VoteAST;

MenuAST : "Menu" name=STRING "{"
      (parts+=PartAST)*
"}";

InfoAST : name=STRING ":" text=STRING;

VoteAST : "Vote" name=STRING
      (choices+=ChoiceAST)*;

ChoiceAST : "-" name=STRING;
```

Figura 10 Gramática

Com essa gramática, o Xtext gera um editor com suporte a code completion, folding, cross referencing, syntax highligting e também um outline da estrutura. Além do editor textual, o

Xtext gera um metamodelo que corresponde à árvore sintática abstrata (AST) da gramática. A DSL na Figura 11 mostra a definição de uma simples aplicação de TV interativa.

```
"World Cup 2010"
Menu "World Cup" {

      Vote "Who will win?"
            - "Estonia"
            - "Lituania"

      Menu "Teams" {
            "England": "England"
            "Scotland": "Scotland"
            "Wales": "Wales"
      }

      "World Cup Trivia":
      "Trivia text goes here..."
}
```

Figura 11 Aplicação TV interativa

O oAW também possui uma outra linguagem, a Xtend, que pode ser usada de diversas formas. Seu principal uso é para transformação de modelos, no entanto pode ser usada para modificação de modelos ou a extensão de um metamodelo. O script da Figura 12 mostra a transformação da DSL em uma DSL mais completa compatível com o editor GMF (Graphical Modeling Framework). É a partir deste artefato que a geração de código vai ser guiada.

```
import itv;

extension org::openarchitectureware::util::stdlib::io;

create dslmetamodel::Application transform (ApplicationAST ast):
      setName (ast.name) ->
      setMain (ast.mainMenu.transform ());
      transform (PartAST ast):
      syserr (ast, "unsupported kind of Part");

create dslmetamodel::Menu transform (MenuAST ast):
      setName (ast.name) ->
      setOptions (ast.parts.transform ());

create dslmetamodel::Info transform (InfoAST ast):
      setName (ast.name) ->
      setText (ast.text);

create dslmetamodel::Vote transform (VoteAST ast):
```

```
      setName (ast.name) ->
      setChoices (ast.choices.transform ());

create dslmetamodel::Choice transform (ChoiceAST ast)
      setName (ast.name);
```

Figura 12 Transformação de modelo com Xtend

Também é possível executar validações de modelos com a linguagem Check, pois os modelos podem estar semanticamente inválidos apesar de estarem sintaticamente corretos. O oAW provê uma DSL específica para restrições que pode ser aplicada em um procedimento em separado. A Figura 13 demonstra um exemplo para a aplicação de TV interativa.

```
import itv;

context Vote WARNING
      "Must have at least two choices..." : this.choices.size>=2;

context Vote ERROR
      "Duplicate choice '"+name+"'" : eContainer.eContents.
          typeSelect(VoteAST).select(e|e.name==name).size==1;

context Part ERROR
      "names must have three chars or more": name.length >= 3;
```

Figura 13 Exemplo da linguagem Check na aplicação de TV interativa

Toda restrição aponta a que tipo de elemento ela está se referindo e é seguida do tipo de severidade de uma validação falha: ERRO ou WARNING. Em seguida, para cada restrição é definida uma mensagem explicativa para orientar os desenvolvedores (HAASE et al., 2007).

Após serem executadas as transformações e aplicada as validações, é chegada a hora da geração de código, que é feita a partir de uma linguagem, também integrada ao oAW. A linguagem Xpand é especifica para definição de templates textuais. É possível gerar qualquer tipo de documento textual com Xpand. A Figura 14 ilustra um template para geração de um documento XML para ser usado na aplicação de TV interativa.

```
«IMPORT itv»

«EXTENSION org::openarchitectureware::util::stdlib::io»

«DEFINE root FOR Application»
  «FILE name+".xml" xml»
    <?xml version="1.0" encoding="utf-8">
      <TVApp name="«name»">
         «EXPAND part FOR main»
      </TVApp>
  «ENDFILE»
«ENDDEFINE»

«DEFINE part FOR Part»
  «syserr (this, "undefined kind of Part")»
«ENDDEFINE»

«DEFINE part FOR Menu»
  <Menu name="«name»">
    «EXPAND part FOREACH options»
</Menu>
«ENDDEFINE»

«DEFINE part FOR Info»
  <Text name="«name»">
    «text»
  </Text>
«ENDDEFINE»

«DEFINE part FOR Vote»
  <Vote name="«name»">
    «FOREACH choices AS c»
      <Choice name="«c.name»"/>
    «ENDFOREACH»
  </Vote>
«ENDDEFINE»
```

Figura 14 Exemplo de template usando a linguagem Xpand

Outro recurso presente no oAW é o uso de classes Java para implementar operações de auxilio para ser usado junto com o Xpand e Xtend. O exemplo da TV interativa é bastante simples e o uso de tal recurso não se faz necessário. Outra característica do Xpand é que os arquivos gerados podem apresentar uma estrutura desorganizada devido a quebras de linhas e espaços. Para solucionar tal problema, é possível definir no workflow a aplicação de um beautifier de código que organiza o texto, deixando o código num formato mais bem estruturado com a correta endentação e sem quebras de linhas desnecessárias.

O próximo passo, que finaliza o processo de criação de uma ferramenta DSL e integra todos os outros componentes previamente mostrados, é o workflow de execução. Este workflow é definido em um arquivo XML e tem o objetivo de guiar a seqüência de execução de cada componente. O workflow da figura 15 define a seqüência de execução para criar a aplicação da TV interativa.

```
<workflow>

  <component id="read" class="oaw.emf.XmiReader">
    <outputSlot value="model"/>
    <firstElementOnly value="true"/>
    <modelFile value="models/modelFile.xmi"/>
  </component>

  <component class="oaw.check.CheckComponent">
    <metaModel id="mm" class="org.openarchitectureware.
          type.emf.EmfMetaModel">
         <metaModelFile value="itv.ecore"/>
      </metaModel>
    <checkFile value="itvConstraints"/>
    <emfAllChildrenSlot value="model"/>
  </component>

  <component id="gen" class="oaw.xpand2.Generator" kipOnErrors="true">
    <metaModel idRef="mm"/>
    <outlet path="src-gen" fileEncoding="iso-8859-1"/>
    <outlet name="xml" path="src-gen" fileEncoding="utf-8" />
      <expand  value="templates::xml::root FOR model"/>
    <beautifier class="oaw.xpand2.output.XmlBeautifier"/>
  </component>

</workflow>
```

Figura 15 Workflow da aplicação de TV interativa

O arquivo XML gerado por esse workflow, tendo como entrada o modelo apresentado, será igual ao XML gerado pelo Visual Studio DSL Tools mostrado no exemplo anterior.

3.3 Estudo comparativo

	Visual Studio DSL Tools	openArchitectureWare
Portabilidade	DSL Tools vem integrado ao Visual Studio não permitindo o uso independente da ferramenta	O oAW pode ser usado para desenvolver para plataformas diferentes
Custo	Proprietário da Microsoft	Software aberto
Flexibilidade	Não permite a integração com outras tecnologias	Permite a integração com diversas tecnologias da plataforma JavaEE
Modelagem	Possui um ambiente de modelagem gráfico bastante sofisticado	Possui um ambiente de modelagem gráfica, no entanto para este estudo foi adotada técnica de modelagem textual
Usabilidade	Necessita de muita configuração detalhada, sendo necessário um estudo mais profundo da ferramenta.	Apenas com a criação dos componentes, sem muita configuração, é possível criar e rodar a aplicação.

Quadro 1 Estudo comparativo

4 ORIGEM, CONCEITO E APLICAÇÃO DO PROJETO WALE (WEB APPLICATION LANGUAGE ENGINE)

O projeto WALE surgiu a partir de um plano da empresa Estalo Consultoria e Automação de migrar do ERP, denominado Fênix, comercializado pela empresa, desenvolvido em Delphi para a tecnologia Java. Como o Fenix é um sistema muito grande seria necessário adotar uma estratégia que pudesse otimizar este processo, pois simplesmente traduzir o código para Java implicaria um alto custo, uma vez que demandaria uma quantidade de tempo para ser concluído. Para atingir este objetivo, foi criada uma frente Java de pesquisa para estudar tecnologias que auxiliassem e acelerassem o processo de migração. O Sculptor foi a ferramenta mais adequada para solucionar o problema.

4.1 Sculptor

O Sculptor é uma ferramenta que permite gerar uma aplicação a partir de uma modelagem em uma DSL, na qual são definidos componentes como entidades, repositórios e serviços. O Sculptor é uma ferramenta interessante, permitindo criar aplicações rapidamente, dado que grande parte das configurações para implantação está pronta. O trecho a seguir é uma definição extraída do site da Fornax-Platform, que desenvolve e mantêm o Sculptor:

> Sculptor é uma simples e poderosa plataforma de geração de código, que prove um rápido inicio ao desenvolvimento orientado a modelos. Ao usar o Sculptor você pode focar no domínio do negócio, ao invés dos detalhes técnicos. Você pode utilizar conceitos de Design Orientado a Domínio (DDD) em uma DSL textual.
>
> A partir da DSL textual o Sculptor gera código Java da alta qualidade e configuração usando o openArchitectureWare. O código gerado é baseado em conhecidos frameworks como Spring Framework, Spring Web Flow, JSF, Hibernate e Java EE. (KAMANN, 2008) O Sculptor faz parte do Fornax Platform que é uma plataforma aberta especifica para o desenvolvimento de ferramentas relacionadas ao Desenvolvimento de Software Orientado a Modelos, baseado no framework de geração oAW. Alguns dos componentes utilizados pela Fornax-Platform são o Subversion (controle de versionamento), o Eclipse (IDE de desenvolvimento), o Maven2 (gerenciamento de buil), o Confluence (wiki de colaboração) e o JIRA (ferramenta de tracking) (KAMANN, 2008).

O desenvolvimento de uma aplicação comercial utilizando o Sculptor resultou em um ganho considerável de produtividade. No entanto, durante o processo concluiu-se que para os projetos seguintes seria necessário que a ferramenta oferecesse mais recursos. A partir

dessa perspectiva, foi decidido criar uma nova ferramenta baseada na arquitetura do Sculptor, acrescida de características especificas para solucionar os problemas existentes nos projetos da empresa. Esta nova ferramenta denominou-se Web Aplication Language Engine (WALE).

4.2 WALE

O WALE é uma ferramenta que possibilita o desenvolvimento de sistemas baseado em técnicas de desenvolvimento orientado a modelos para o domínio web. A escolha do domínio da web partiu das deficiências da estrutura atual do Fênix, que é totalmente desktop, o que resulta em um grande esforço no processo de implantação devido a necessidade de instalar uma versão em cada estação de trabalho. No caso de implantar um sistema distribuído web, basta instalar a aplicação em um servidor de aplicação web, e tirar proveito da infra-estrutura de rede da empresa. O domínio do WALE é focado em sistemas configurados para a plataforma JavaEE junto com vários frameworks de suporte para esta plataforma. Um dos benefícios do WALE é a possibilidade de se ter um sistema funcional, configurado e rodando num ambiente web já na criação do projeto.

4.2.1 Plugin WALE

O WALE foi desenvolvido para ser um plugin do eclipse, dessa forma facilitando o trabalho do desenvolvedor, que só precisa instalá-lo para começar a usar. O desenvolvimento do plugin é baseado em quatro projetos desenvolvidos no eclipse: wale.dsl, wale.dsl.editor, wale.dsl.generator e wale.dsl.metamodel.

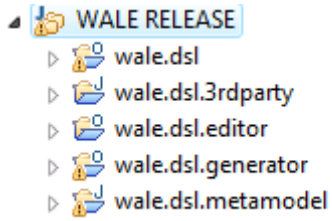

Figura 16 Projeto WALE

O projeto wale.dsl é responsável pela definição da gramática da DSL do WALE, que é definida no arquivo “wale.xtxt”. A gramática define a sintaxe e a semântica da DSL em uma linguagem na forma EBNF. Esta gramática é mapeada pelo oAW em um arquivo ecore que pode ser visualizado graficamente num formato de arvore, como mostra a Figura 17.

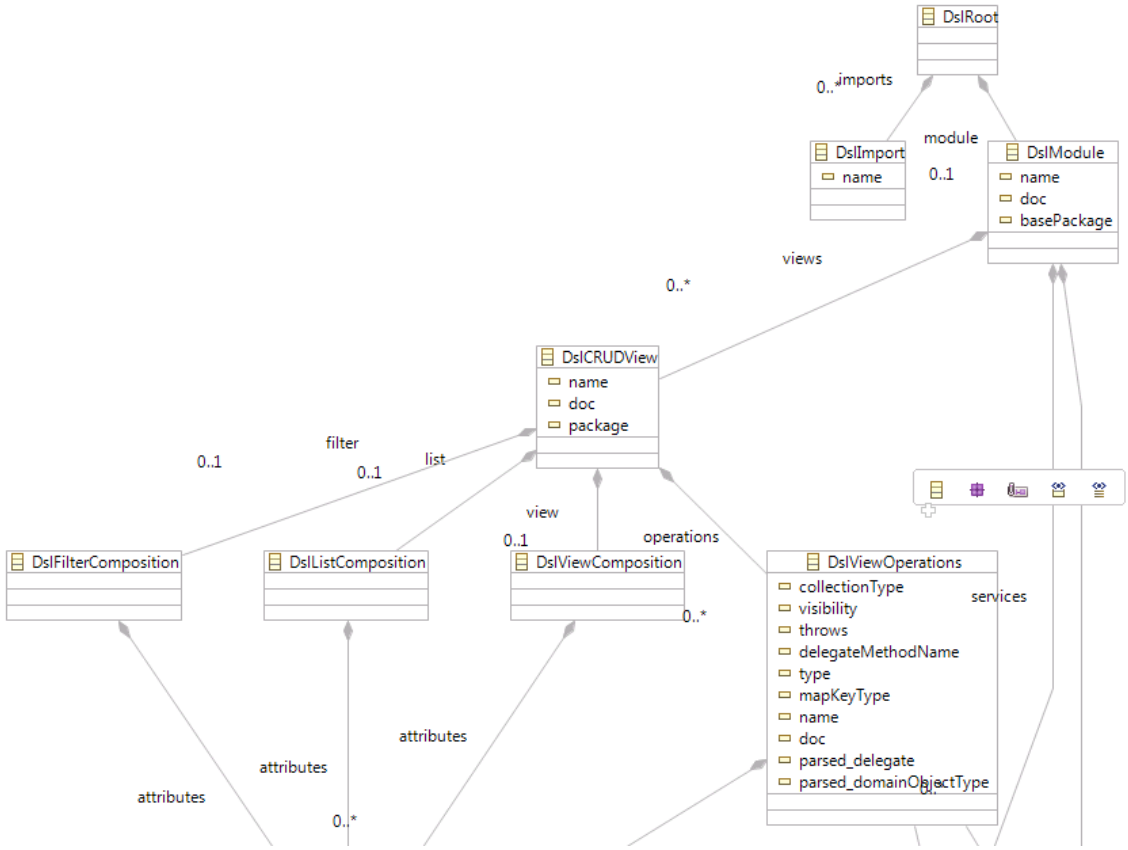

Figura 17 Representação gráfica da gramática do WALE

O projeto wale.dsl.editor é responsável pelo editor textual da DSL, a partir do qual os projetos WALE podem ser modelados textualmente. O editor textual é bastante sofisticado e possui recursos de edição como cross referincig, code assist, highlight e um outline dos componentes criados.

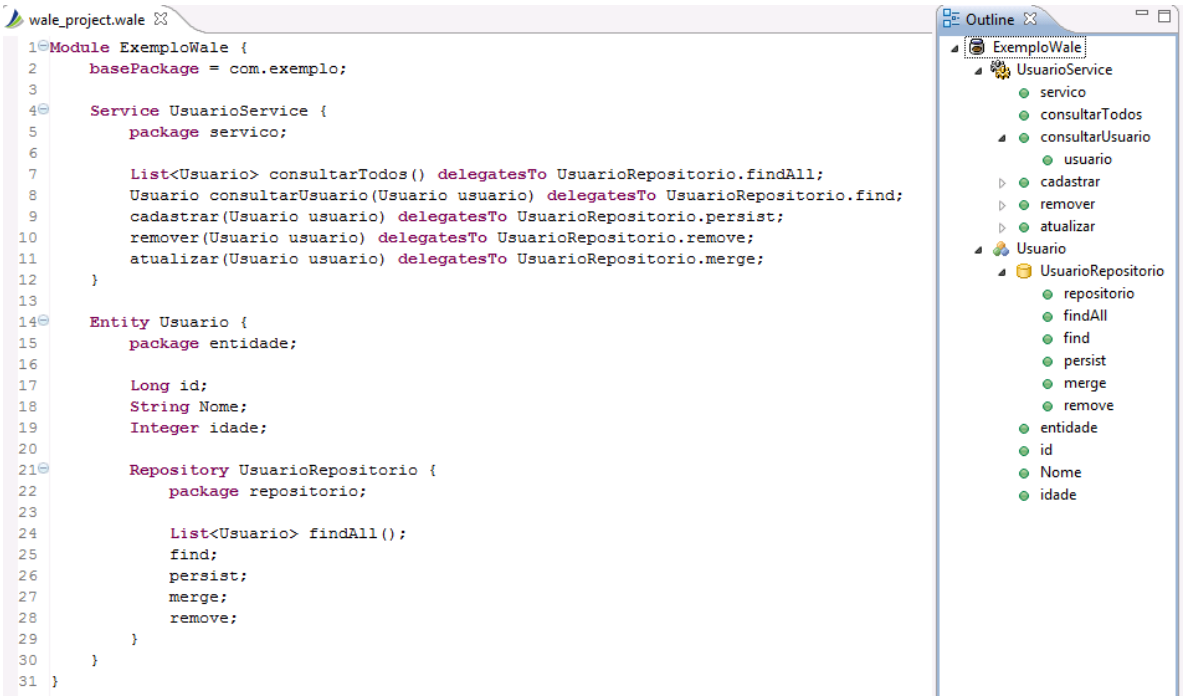

Figura 18 Editor WALE DSL e outline

O projeto wale.dsl.generator é responsável pelo workflow do projeto, ou seja, é nele que é configurada a seqüência de geração e validação de código. Este projeto é o mais importante e é o que concentra a maior carga de trabalho no desenvolvimento do WALE. É no wale.dsl.generator que estão os arquivos Xpand com extensão "xpt". São nestes arquivos que são definidos os templates de geração de código. Há também os arquivos Xtends, com extensão "ext", que estendem os arquivos xpt auxiliando em tarefas que a linguagem Xpand não resolve. A linguagem Xtend também é usada nas transformações entre modelos. Alem destes dois tipos de arquivo essenciais, existem arquivos Java que também tem o propósito de auxiliar em tarefas onde as linguagens Xpand e Xtend não resolvem. Outro arquivo importante, senão o mais importante, é o generator.oaw. Este arquivo é o arquivo de workflow que contém o passo a passo da execução para geração de código.

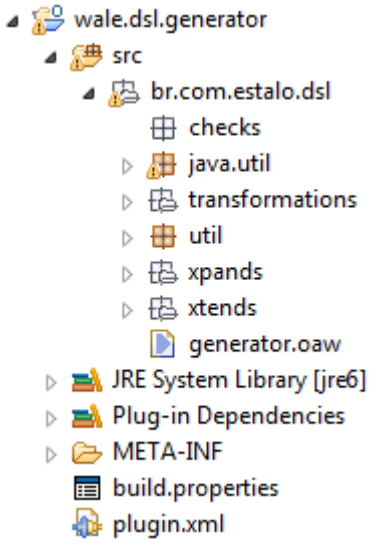

Figura 19 Projeto wale.dsl.generator

A base da geração de código está nos templates de arquivos que são escritos na linguagem Xpand. A Figura 20 é o trecho de um código de template para a camada de serviço.

```
«IMPORT metamodel»

«EXTENSION br::com::estalo::dsl::xtends::service»
«EXTENSION br::com::estalo::dsl::xtends::shared»

«DEFINE service FOR DslService»
    «EXPAND serviceBaseInterface»
    «EXPAND serviceInterface»
    «EXPAND serviceBase»
    «EXPAND serviceImpl»
    «EXPAND droolsRules»
«ENDDEFINE»

«DEFINE serviceBaseInterface FOR DslService»
    «FILE  transformPackageString(getFullPackage(this)) + "/" + getServiceBaseIntfName() + ".java"»
        «getJavaLicenseHeader()»
        package «getFullPackage(this)»;

        public interface «getServiceBaseIntfName()» extends br.com.estalo.service.ServiceLocal {

            // Constants ----------------------------------------------------

            // Public -------------------------------------------------------

            «EXPAND interfaceMethod FOREACH operations.select(op|op.isPublicVisibility() && op.delegate != null )»

        }
    «ENDFILE»
«ENDDEFINE»

«DEFINE serviceBase FOR DslService»
    «FILE transformPackageString(getFullPackage(this)) + "/" + getServiceBaseName() +".java"»
        «getJavaLicenseHeader()»
        package «getFullPackage(this)»;

        @br.com.estalo.service.annotations.Rule
        public class «getServiceBaseName()» extends br.com.estalo.service.AbstractService implements «getServiceBaseIntfName()» {

            // Constants ----------------------------------------------------

            // Attributes ---------------------------------------------------

            «EXPAND attributes FOR this»

            // Static -------------------------------------------------------

            // Constructors -------------------------------------------------

            public «getServiceBaseName()»() {
                super();
            }

            // Public -------------------------------------------------------

            «EXPAND repositoriesGetSet FOR this»

            // Z implementation ---------------------------------------------

            «EXPAND serviceBaseClassMethods FOREACH operations.select(op|op.delegate != null)»

            // Y overrides --------------------------------------------------

            // Package protected --------------------------------------------

            // Protected ----------------------------------------------------

            // Private ------------------------------------------------------

            // Inner classes ------------------------------------------------

        }
    «ENDFILE»
«ENDDEFINE»
```

Figura 20 Arquivo de definição do template dos serviços, service.xpt

Por ultimo o wale.dsl.metamodel, é o projeto que contém a representação dá gramática em um metamodelo, o que auxilia nas transformações e geração de código. No projeto wale.dsl

existe um arquivo “xtxt” que contém a definição da gramática da DSL, e a partir dele um arquivo ecore é gerado para representar a gramática de forma estruturada. O projeto wale.dsl.metamodel faz uso deste arquivo ecore para gerar uma representação em código Java do modelo da gramática, o que permite a manipulação dos componentes da gramática nos arquivos de template. O generator.oaw é o arquivo que é executado quando o usuário quer gerar o código da aplicação a partir da DSL.

4.2.2 Arquitetura do projeto WALE

A arquitetura do WALE é bastante simples e não difere muito das arquiteturas tradicionais em camadas. A disposição das camadas é divida em interface gráfica (GUI), que neste caso é web, ou seja, pode ser feita com arquivos HTML (HyperText Markup Language), no entanto é usado o modelo XHTML (eXtensible Markup Language) que é uma extensão do HTML pois integra recursos de XML permitindo que possa ser interpretado por qualquer dispositivo, independente de plataforma. Neste caso usa-se XHTML como HTML, porém fazendo uso de diversas tags especificas de frameworks para interface gráfica, que potencializam e adicionam recursos dinâmicos tornando a usabilidade da web melhor e mais poderosa. A camada abaixo da GUI é a dos controladores que coordena todos os recursos utilizados pela interface gráfica, e faz a ligação destes com a camada de serviço. A camada de serviço é a camada de negócio onde são validadas as regras de negócio. A camada de serviço é especial, pois faz uso de um framework chamado JBoss Rules, ou Drools, que é uma engine de regras open source todo desenvolvido em Java. Com o Drools é possível escrever regras de negócio através de uma DSL simples que contém duas condicionais when e then, ou, traduzindo, quando e então. Essas condicionais funcionam da seguinte forma: quando algo acontecer, então faça algo. Dessa forma é possível validar de forma simples regras de negócio complexas. A camada de serviço tem acesso à camada de repositório, que é responsável por acessar os dados em algum tipo de armazenamento, seja ele um arquivo, banco de dados, ou outro. Outra camada é a camada de dados que representa as entidades mapeadas do banco para uso em toda a aplicação. O diagrama a seguir mostra a arquitetura completa do WALE

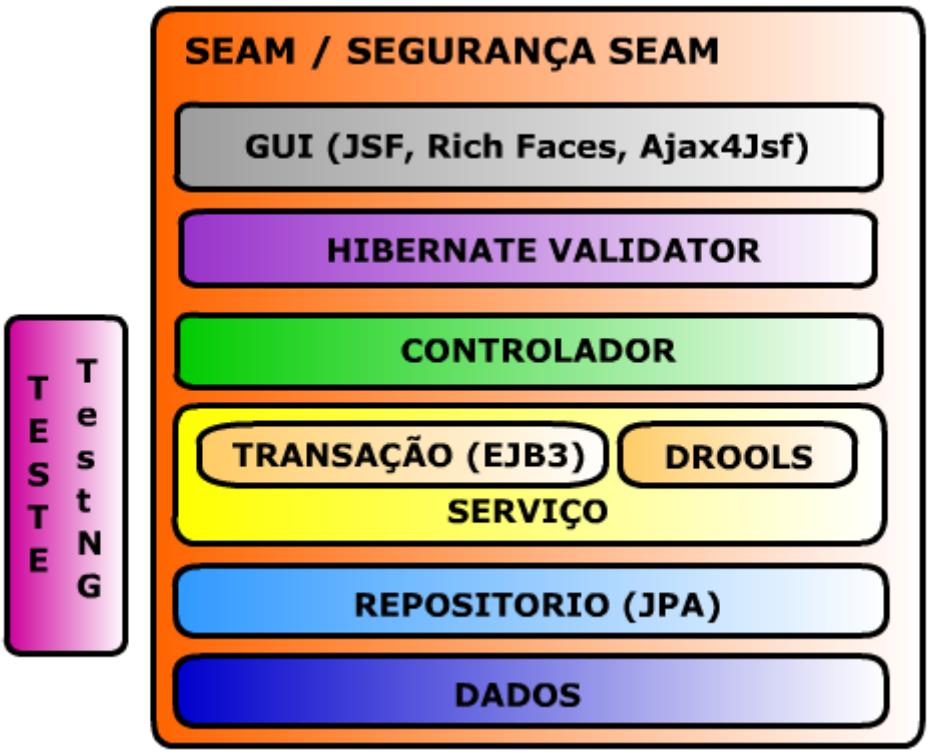

Figura 21 Diagrama da arquitetura do WALE

4.2.3 Estrutura de Frameworks

A ferramenta WALE trabalha com varias tecnologias diferentes, cada uma com uma função especifica no projeto. Estas tecnologias são frameworks e APIs amplamente usadas em projetos de software, pois fornecem recursos e soluções para problemas recorrentes. Os frameworks usados agem em diversos domínios de uma aplicação, desde a interface gráfica, à segurança, a regras e ao acesso a dados. O WALE faz uso do Java Server Faces (JSF), Rich Faces e Ajax for JSF para interface gráfica, o Drools para as regras, o TestNG para testes unitários, o Hibernate junto com Java Persistece API (JPA) para o acesso a dados. Outro framework importante é o Seam que é responsável pela integração entre JSF e Enterprise Java Beans (EJB) e também pela segurança da aplicação. O EJB é uma arquitetura de componentes para JavaEE do lado do servidor que permite o desenvolvimento ágil e simplificado de aplicações distribuídas, transacionais e portáveis. O WALE usa o EJB3 como padrão, pois essa versão é totalmente nova e simplificada com

relação às versões anteriores. Em projetos comuns, a configuração e incorporação dessas tecnologias levam tempo e é necessário pessoas com conhecimento profundo em cada uma para deixar o ambiente configurado e pronto para o desenvolvimento da aplicação. O WALE resolve este problema ao criar as aplicações com o ambiente de desenvolvimento totalmente configurado para suportar todos estas tecnologias, poupando os desenvolvedores da tarefa árdua de incorporação de cada framework.

Frameworks

- JSF: Java Server Faces foi criado para facilitar o desenvolvimento de interfaces gráficas em aplicações web na plataforma JavaEE. JSF trás consigo uma gama de componentes que agilizam o processo de criação de uma interface gráfica, além de oferecer recursos para ligar estes componentes visuais ao código da aplicação.
- RichFaces: Rich Faces é uma biblioteca de componentes para JSF e um framework para a integração da tecnologia AJAX (Asynchronous Javasript And Xml) com aplicações JavaEE. O Rich Faces agiliza o desenvolvimento por fornecer componentes prontos para serem usados.
- Ajax for JSF: Ajax faz uso das tecnologias de JavaScript e XML para tornar as páginas mais dinâmicas e interativas com o usuário. Sua característica principal é o uso de solicitações de informações assíncronas o que permite a atualização de componentes e informações independentemente, sem necessitar fazer uma solicitação completa da página. O Ajax para JSF é uma integração das técnicas de Ajax com JSF facilitando o desenvolvimento de aplicações baseadas em JSF.
- Seam: É uma plataforma de desenvolvimento para aplicações web em Java, que integra tecnologias como Ajax, JSF, JPA e EJB3 em um único ambiente. O Seam simplifica o desenvolvimento de aplicações tanto no nível arquitetural quanto no das APIs. Isso tudo é devido ao uso de anotações que simplificam o desenvolvimento e reduzem consideravelmente o uso de XML para configuração (SEAM, 2008).
- EJB3: Enterprise Java Beans 3.0 (EJB3) é uma tecnologia de componentes arquiteturais do lado do servidor para JavaEE que permite o desenvolvimento

rápido e simplificado de aplicações distribuídas, transacionais, seguras e portáveis.

- Drools: É um framework de regras de negócio que permite ao desenvolvedor criar regras de forma declarativa, sem se preocupar em dizer "como fazer", mas sim "o que fazer". As regras são escritas em uma linguagem de sintaxe bem simples, com apenas duas condicionais "when" e "then". Na condição when o desenvolvedor define regras e restrições para ativar a condição then. A condição then define o que deve ser feito após a avaliação das regras na cláusula then. O Drools também permite a manipulação de objetos e a execução de métodos fora de seu contexto.
- JPA: É um framework específico para manipulação de dados na camada de persistência. O JPA foi definido como parte do EJB3 e trás consigo varias vantagens e simplificações. O JPA foi criado para ser o padrão de tecnologia para persistência, baseado em tecnologias consolidadas neste domínio como Hibernate, TopLink e JDO.
- Hibernate: O Hibernate é um framework de persistência assim como o JPA, no entanto no projeto WALE é usado apenas o recurso de validação de dados. O Hibernate Validator atua nas entidades para checar qualquer violação de restrição relacionada a uma dada entidade.
- TestNG: É um framework para testes para simplificar uma gama de necessidades de teste. O WALE usa o TestNG ao invés do JUnit pois é mais completo e possui mais funcionalidades. O TestNG funciona em três passos: 1) definir lógica de negócio do teste; 2) adicionar anotações do TestNG; 3) rodar o TestNG.
- Ant: É uma ferramenta baseada em Java pra fazer o build ou compilação, e deploy da aplicação, dentre varias outras funções possíveis de fazer com o Ant.

4.2.4 Usando o WALE

Por ser um plugin é possível criar um projeto WALE através do menu de "novo projeto". O usuário irá seguir um breve wizard, informando o nome do projeto, como pode ser

visualizado na Figura 22. Após esta etapa, um projeto, com toda a estrutura necessária para o desenvolvimento de uma nova aplicação, é criado. Esta estrutura inicial ainda é muito simples, e só fornece os recursos básicos de uma aplicação, com uma interface gráfica simples contendo a página inicial e uma página de erro, arquivos de configuração na pasta "src-res", e os arquivos de modelagem e de workflow contidos na pasta "src". Na raiz há alguns arquivos de propriedades e um arquivo de build do ant, responsável por fazer o deploy da aplicação no servidor. Esta disposição do projeto pode ser visualizada na Figura 23.

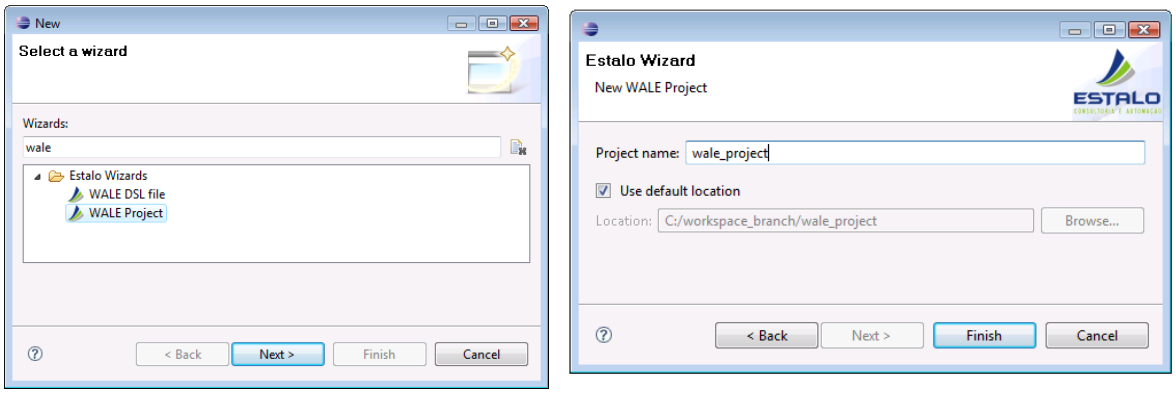

Figura 22 Wizard do WALE

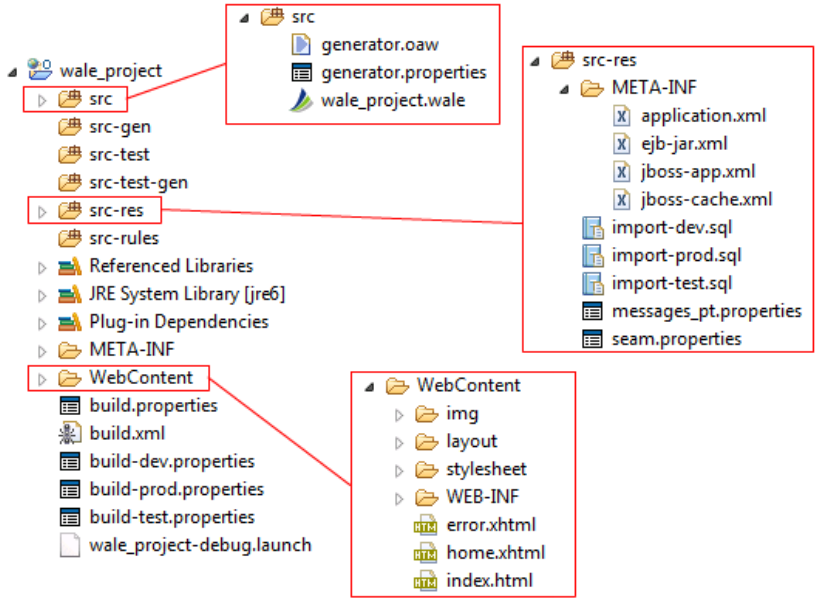

Figura 23 Estrutura do projeto WALE

Depois de criado um novo projeto WALE o desenvolvedor pode começar a modelar uma aplicação. A modelagem é feita através de uma DSL textual no arquivo com extensão "wale" contido na pasta "src". Um exemplo de modelo pode ser visto na Figura 24. Quando a modelagem estiver pronta é hora de gerar o código e os artefatos do sistema. Para isso, o desenvolvedor deve executar o arquivo de workflow "generator.oaw". A execução deste workflow irá gerar os códigos de serviço, repositório, entidades e testes, assim como os artefatos de regras do Drools e arquivos de propriedades que contém parâmetros para o funcionamento da aplicação, Figuras 25 e 26.

```
Module ExemploWale {
	basePackage = com.exemplo;

	Service UsuarioService {
		package = servico;

		List<Usuario> consultarTodos() delegatesTo
				UsuarioRepositorio.findAll;
		List<Usuario> consultarUsuario(Usuario usuario) delegatesTo
				UsuarioRepositorio.find;
		cadastrar(Usuario usuario) delegatesTo
				UsuarioRepositorio.persist;
		remover(Usuario usuario) delegatesTo
				UsuarioRepositorio.remove;
		Usuario atualizar(Usuario usuario) delegatesTo
				UsuarioRepositorio.merge;
	}

	Entity Usuario {
		package = entidade;

		Long id key(generated);
		String nome;
		Integer idade;

		Repository UsuarioRepositorio {
			package = repositorio;

			List<Usuario> findAll();
			find;
			persist;
			remove;
			merge;
		}
	}

	ScreenCRUD UsuarioCRUD {
		package = gui;

		entity = Usuario;
```

```
            create = UsuarioService.cadastrar;
            delete = UsuarioService.remover;
            read = UsuarioService.consultarUsuario;
            update = UsuarioService.atualizar;
        }
}
```

Figura 24 Exemplo de modelo na DSL textual do WALE

Os artefatos gerados são distribuídos em diferentes pastas dentro do projeto. Algumas pastas são reservadas para os arquivos que serão gerados sempre que o workflow for executado. Estas pastas são identificadas por sufixos “gen”, como em “src-gen”. Existem duas pastas a “src-gen” e a “src-test-gen”, que contém os códigos que representam as classes base. As classes base contem o conteúdo definido no modelo, portanto nunca devem ser alteradas, apenas quando há uma alteração no modelo. Outro tipo de classes geradas são as classes que herdam destas classes base. Já estas classes podem ser alteradas, pois são geradas apenas uma vez. É nelas que o desenvolvedor acrescenta detalhes de implementação específicos para o sistema. Essas classes ficam distribuídas em pacotes, definidos no modelo, dentro das pastas “src” e “src-test”. Outros artefatos gerados são as regras, que residem na pasta “src-rules”. As regras são geradas de acordo com os serviços. Para cada método de um serviço existe um arquivo de regra correspondente, dessa forma é possível associar cada regra a cada método da camada de negócio do sistema.

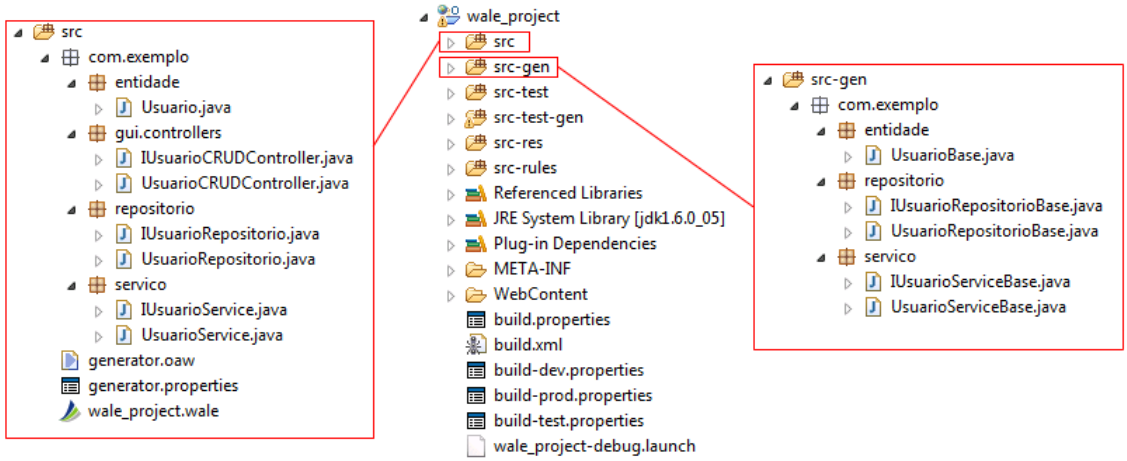

Figura 25 Código gerando nas pastas src e src-gen.

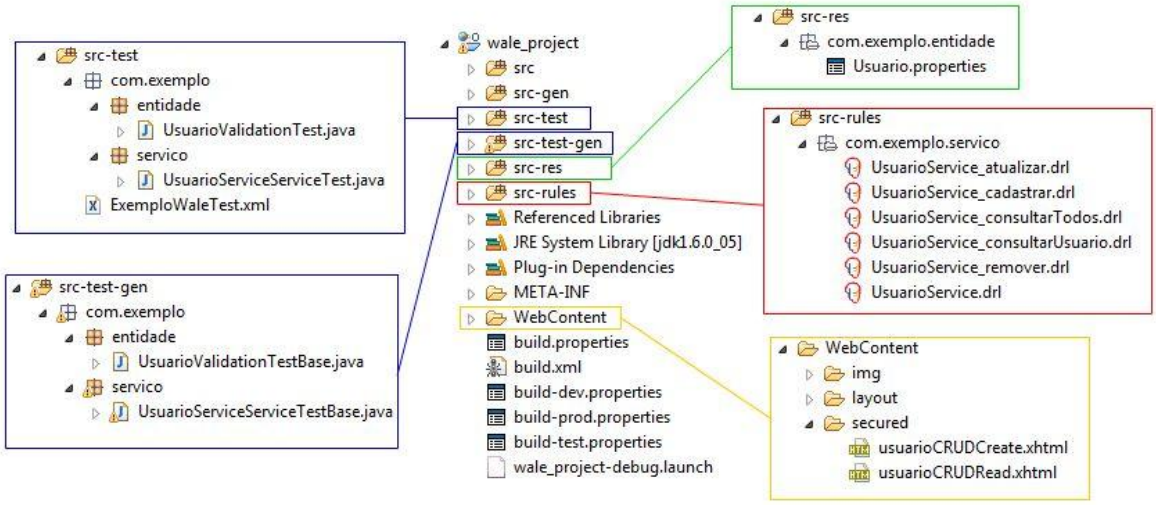

Figura 26 Artefatos gerados de teste, propriedades, regras e de interface gráfica.

O uso da ferramenta WALE é bastante simples e não requer um profundo conhecimento de todas as tecnologias utilizadas. Grande parte to trabalho reside na definição de um modelo. Com o modelo pronto basta gerar a aplicação e trabalhar nos detalhes relevantes, como a definição da lógica de acesso a dados e as regras de negócio. Na faze atual do WALE a geração da camada de interface gráfica é bastante simples, apenas gerando simples funções de CRUD. Para o funcionamento do WALE com um banco deve-se configurar uma conexão nos arquivos “build-dev.properties”, “build-prod.properties” ou “build-test.properties” dependendo de como o desenvolvedor irá usar a aplicação, seja para desenvolvimento, produção ou testes. Após a primeira geração já é possível compilar o código e fazer o deploy da aplicação no servidor através do arquivo de build do ant, “build.xml”. Após o build é só iniciar o servidor e testar o sistema em um navegador. Neste exemplo é possível visualizar a página inicial, uma pagina de cadastro e uma de consulta para a entidade Usuario, como mostra as Figuras 27, 28 e 29.

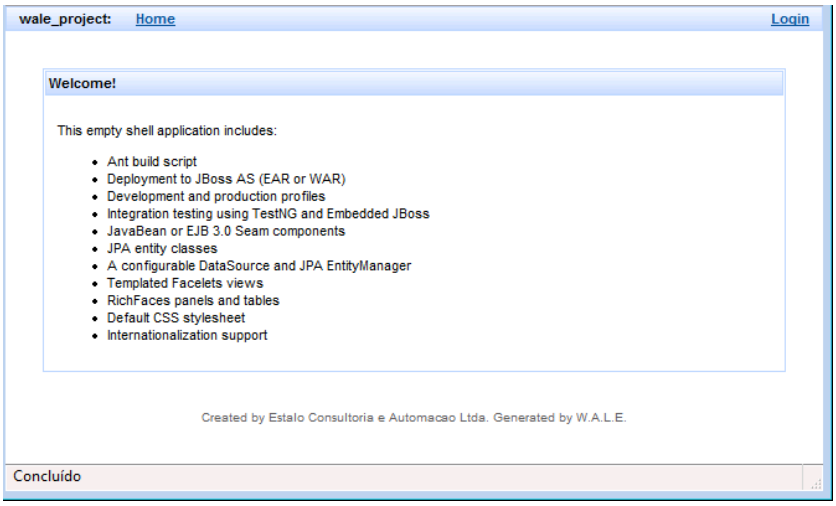

Figura 27 Página inicial do WALE

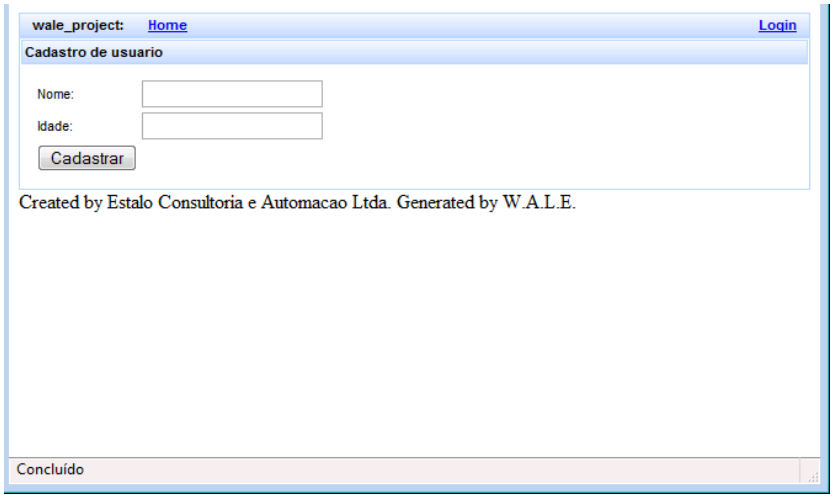

Figura 28 Página de cadastro da entidade Usuario

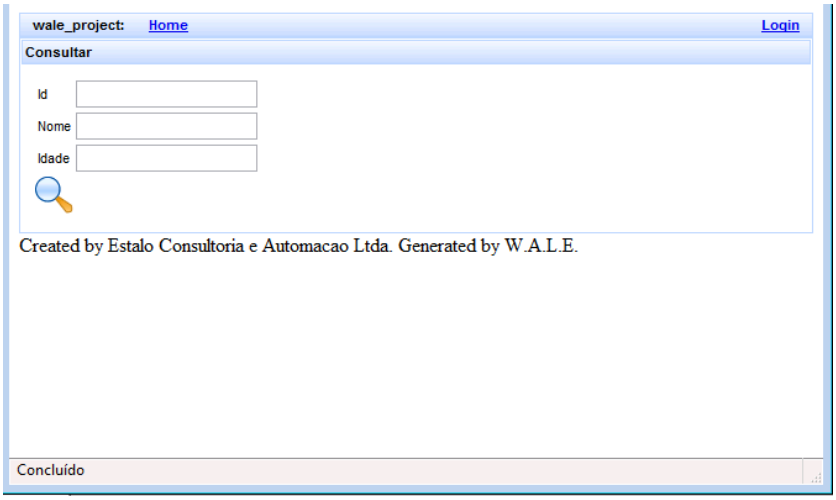

Figura 29 Página de consulta da entidade Usuario

5 CONCLUSÃO

Diante do trabalho apresentado há algumas considerações que podem ser ressaltadas com relação às soluções desenvolvidas. As técnicas de desenvolvimento orientado a modelos trazem consigo grandes vantagens para o ambiente de produção de software. No entanto pode-se notar que a aplicação de tais técnicas depende muito do objetivo final e o propósito a qual será aplicada. Deve-se atentar que a criação de ferramentas para o desenvolvimento de uma única aplicação é inviável, devido ao fato de que é custoso criar tais ferramentas, o que implicaria em um trabalho desnecessário, já que a ferramenta cairia em desuso depois da conclusão de uma aplicação. Com base nisto pode-se concluir que em um ambiente onde são desenvolvidos softwares que atuam em um mesmo domínio é plausível a criação de uma ferramenta para auxiliar no processo de desenvolvimento, haja vista o ganho de produtividade devido ao fato de reduzir a quantidade de retrabalho e de poder gerar grande parte dos artefatos através de transformações de modelos e geração de código, promovendo assim o reuso de componentes.

Com relação ao estudo de caso é possível destacar algumas características que levaram a escolha da ferramenta mais adequada para criação de uma ferramenta de desenvolvimento orientado a domínio. A relação de custo, flexibilidade e usabilidade foram levadas em consideração para tal decisão. A ferramenta DSL Tools apresenta um agravante no contexto do custo, por ser uma ferramenta proprietária, e também na flexibilidade por não permitir a extensão e integração de suas funcionalidades. Já o oAW não apresenta custos por ser um framework aberto, e permite uma flexibilidade para estender e integrar seus recursos às diversas tecnologias e plataformas. No quesito de usabilidade pode-se ter uma avaliação relativa, pois depende do contexto onde serão usadas. A adoção de uma ferramenta com suporte à modelagem gráfica tem benefícios evidentes por permitir que se monte uma aplicação através de componentes visual, tornando a interação do usuário com a ferramenta mais intuitiva e agradável. Por outro lado, o uso de uma ferramenta de modelagem textual tem um apelo maior para usuários desenvolvedores que já estão acostumados com o desenvolvimento em linha de código.

O ambiente de desenvolvimento de aplicações web é bastante complexo por possuir uma infinidade de tecnologias, o que requer pessoas com alto nível de conhecimento para configurar e integrar todos os componentes necessários para o desenvolvimento de um sistema web. Com base nisso a criação da ferramenta WALE apresenta uma abstração da plataforma de implementação, permitindo que os desenvolvedores se concentrem na modelagem da aplicação e nos detalhes essencias do domínio de negócio. O uso de uma ferramenta como esta em empresas de desenvolvimento de software traz benefícios em todas as camadas da empresa, por oferecer um ganho de produtividade, uma redução de custo de manutenção, uma redução de documentação e um ganho na qualidade dos produtos desenvolvidos. O desenvolvimento orientado a modelos é sem dúvida uma técnica que vem mostrando vantagens, no entanto ainda tem um longo caminho para se tornar uma metodologia padrão para o desenvolvimento de software.

6 REFERÊNCIAS

BALASUBRAMANIAN, K. et al. Developing Applications Using Model-Driven Design Environments. v. 39, n. 2, p. 33-40, Fevereiro 2006 2006.

COOK, S. Interactive Television Applications using DSL Tools. In: Model Driven Development Tools Implementers Forum, Tools, 2007.

FOWLER, M. Language workbenches: the killer-app for domain specific languages? , v. 2008, 2005. Disponível em:<http://www.martinfowler.com/articles/languageWorkbench.html>. Acesso em: 30 de out. 2008.

GARDNER, T.; YUSUF, L. Explore model-driven development (MDD) and related approaches: a closer look at model-driven development and other industry initiatives. v. 2008, 2006. Disponível em:<http://www.ibm.com/developerworks/library/ar-mdd3/>. Acesso em: 16 out. 2008.

HAASE, A. et al. Introduction to openArchitectureWare 4.1.2 5 mai., 2007.

KAMANN, T. Sculptor. v. 2008, n. 16/10/2008, 2008. Disponível em:<http://www.fornax-platform.org/cp/display/fornax/Sculptor+(CSC)>. Acesso em: 02 de nov. 2008.

LANGLOIS, B.; EXERTIER, D.; BONNET, S. Performance improvement of MDD tools In: INTERNATIONAL ENTREPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS, Washington, DC, **Proceedings ...** Washington, DC: IEEE Computer Society, 2006. p.19.

OMG. **MDA Guide Version 1.0.1**. 2003.

OMG. OMG Model Driven Architecture. v. 2008, n. 16/10/2008, 2008. Disponível em:<http://www.omg.org/mda/>. Acesso em: 17 de abr. 2008.

OPENARCHITECTUREWARE. n. 19/10/2008, 2008. Disponível em:<**Erro! A referência de hiperlink não é válida.**.

SCHMIDT, D. C. Guest Editor's Introduction: Model-Driven Engineering. v. Volume 39, n. Issue 2, p. 25 - 31 Fevereiro 2006 2006.

SEAM. The Seam Framework - Next generation enterprise Java development. v. 2008, 2008. Disponível em:<http://www.seamframework.org/>. Acesso em: 15 nov. 2008.

STEVEN KELLY, J.-P. T. **Domain-specif modeling: enabling full code generation**. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

WHITFIELD, A. Case Study: Model Driven Architecture for Avionics Systems Development 5/11/2007 2007.

YUSUF, L.; CHESSELL, M.; GARDNER, D. T. Implement model-driven development to increase the business value of your IT system. v. 2008, 2006. Disponível em:<http://www.ibm.com/developerworks/library/ar-mdd1/>.