Evolução de uma Arquitetura de Framework de Aplicação para Sistemas de Informação com Desenvolvimento Dirigido por Modelos

Valdemar Vicente Graciano Neto[1], **Juliano Lopes de Oliveira**[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) - Câmpus Formosa
Rua 64, esq. c/ Rua 11, s/n, Expansão Parque Lago – 73.813-816. Formosa - GO – Brasil

[2]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 - CEP 74001-970 - Goiânia - GO

{vgracianoneto}@gmail.com, juliano@inf.ufg.br

Resumo. *O Grupo de Pesquisa em Engenharia de Software do Instituto de Informática da UFG tem se aplicado no desenvolvimento de um Framework de Aplicação (FA) para Sistemas de Informação (SI) no intuito de obter ganhos advindos do reúso característico deste tipo de artefato na Engenharia de Software que, combinado com conceitos de Engenharia de Software Dirigida por Modelos (ESDM), dão suporte à geração automática de software de SI. Entretanto, o surgimento de novos requisitos explicitou dificuldades em evoluir a arquitetura da versão original desse FA devido a problemas como replicação e espalhamento de código, baixa flexibilidade da arquitetura e, principalmente, a falta de referencial teórico sobre FA associados a ESDM. Isso motivou a seguinte questão de pesquisa: Como estruturar uma arquitetura de um FA dirigido por modelos para SI? Para responder a estas questões, foi conduzida uma pesquisa e foi concebida uma Arquitetura de Software de FA que usa conceitos de ESDM para sintetizar SI. Essa proposta arquitetural constitui uma evolução da primeira e provê uma solução compreensiva para fomentar a geração e reúso de código no desenvolvimento de software de SI.*

Abstract. *The Software Engineering Research Group of the Institute of Informatics in Federal University of Goiás has invested in an Application Framework (AF) for Information Systems (IS) development using Model-Driven Engineering (MDE) concepts to support the automatic generation of IS software. However, new requirements revealed troubles to evolve the first AF architecture version due to problems such as replication and spreading code, low flexibility of the architecture, and especially the lack of theoretical framework on AF associated with MDE. This led to the following research question: How to design an architecture for a model-driven AF for IS? To answer this question, we conducted a survey and we designed an AF architecture that uses MDE to synthesize IS components. This proposal is an architectural evolution of the first and provides a comprehensive solution to foster the generation and reuse of code in the IS software development.*

1. Introdução

O desenvolvimento de Sistemas de Informação (SI) mantém-se como uma atividade cara e complexa. Grandes esforços são demandados em todos os estágios do processo de

desenvolvimento para lidar com a qualidade dos requisitos do cliente dentro de orçamento e prazos reduzidos.

O Grupo de Pesquisa em Engenharia de Software do Instituto de Informática da Universidade Federal de Goiás (INF-UFG) tem investigado a aplicação de técnicas de reúso através de Frameworks de Aplicação associados a conceitos de Engenharia de Software Dirigida por Modelos (ESDM) ou *Model-Driven Engineering* (MDE) [Butler et al. 2002] no desenvolvimento de software para o domínio de Sistemas de Informação (SI) [da Costa 2011, da Costa et al. 2010, Boff 2010, de Almeida 2010, da Silva 2010].

O objetivo desse grupo de pesquisa é desenvolver um Framework de Aplicação (FA) para dar suporte à geração automática de software de SI. A intenção do grupo é prover evidência empírica de que um FA poderia aumentar a produtividade na construção de software de SI, e melhorar a qualidade do produto de software por prover reúso tanto de código quanto de projeto. Espera-se também confirmar que, quando FA são combinados com técnicas de ESDM, esses benefícios possam ser maximizados.

Apesar dos sucessos iniciais atingidos na construção de um FA para SI [Almeida et al. 2009], o surgimento de novos requisitos explicitou dificuldades em evoluir a arquitetura da versão original desse FA devido a problemas como replicação e espalhamento de código, baixa flexibilidade da arquitetura e, principalmente, a falta de referencial teórico sobre FA associados a ESDM [Graciano Neto et al. 2010].

O principal objetivo deste trabalho é responder à seguinte questão de pesquisa: *Como estruturar uma arquitetura de framework de aplicação dirigido por modelos para SI?*

O restante deste artigo, baseado nos resultados de uma dissertação de mestrado [Graciano Neto 2012], descreve resultados e lições aprendidas do trabalho de evolução de uma arquitetura para frameworks de aplicação de SI.

A Seção 2 discute os elementos teóricos de Sistemas de Informação e Frameworks de Aplicação que são necessários para compreender os propósitos deste trabalho. A Seção 3 trata da evolução do framework em termos de novas características arquiteturais e funcionalidades. A Seção 4 apresenta as considerações finais e indica direções para trabalhos futuros.

2. Frameworks de Aplicação e Desenvolvimento Dirigido por Modelos

Esta seção apresenta os pilares teóricos sobre os quais este trabalho foi conduzido. Apresenta conceitos úteis para a compreensão integral do problema e da proposta de solução apresentada.

2.1. Engenharia de Software Dirigida por Modelos

Modelos são artefatos da Engenharia de Software para capturar o conhecimento adquirido durante o processo de desenvolvimento. Eles ajudam analistas a entender problemas complexos e suas soluções potenciais através de abstração. Vários modelos são usados para expressar os conceitos do domínio de conhecimento para o qual o software é construído, e existem modelos específicos para cada fase do processo de desenvolvimento de software.

A Engenharia de Software Dirigida por Modelos (ESDM) [Butler et al. 2002] é um novo modelo prescritivo de processo de desenvolvimento de software derivado da abordagem **MDA** (*Model-Driven Architecture*) para criar software a partir de transformações sucessivas entre os modelos gerados em cada etapa do processo de desenvolvimento de software. É uma abordagem baseada em transformadores de modelos e metamodelos.

Para transformar modelos utiliza-se uma **definição de transformação**, que é um conjunto de regras que descrevem como um modelo em uma linguagem origem pode ser transformado em um modelo em uma linguagem destino [Kleppe et al. 2003]. Segundo a MDA, as regras ou definições de transformação são elementos conectáveis à ferramenta de transformação.

Por ser um framework conceitual, MDA não apresenta propostas arquiteturais para as ferramentas de transformação dirigidas por modelos. Logo, cada iniciativa implementa a conexão de regras de transformação de uma forma diferente. A falta de consenso sobre como aderir à MDA é um fator que motivou o projeto da ferramenta tema desta pesquisa. A próxima seção apresenta conceitos sobre Frameworks de Aplicação.

2.2. Frameworks de Aplicação

Um Framework de Aplicação (FA) é uma aplicação semi-completa, construída como uma coleção organizada de componentes de software reusáveis para facilitar a implementação de aplicações de software customizadas [Fayad and Schmidt 1997] e tem se tornado o padrão *de-facto* para implementar sistemas para negócios [Mailloux 2010].. Tipicamente, esse tipo de ferramenta é apresentado como um conjunto de classes que constitui um esqueleto de um produto de software: ele contém lacunas (ou *hot spots*) que devem ser preenchidas usando código manualmente inserido e específico para o produto [Santos et al.] e alguns pontos de imutabilidade criados intencionalmente para não serem modificados por seus usuários, chamados de *frozen spots* [Markiewicz and de Lucena 2001].

Em geral, frameworks de aplicação não são diretamente executáveis. Para gerar código executável, é preciso instanciar este FA implementando código específico de plataforma para cada *hot spot*. O código definido pelo usuário é chamado por um FA na ocorrência de um dado evento. Isso é conhecido como o *princípio da inversão de controle* associado ao FA [Fowler 2005].

As principais características de um FA são modularidade, reúso, extensibilidade e Inversão de Controle (IC) [Fayad and Schmidt 1997, Fowler 2005].

As características de um FA são influenciadas pelo domínio da aplicação. Logo, construir este tipo de software requer uma análise criteriosa do domínio para compreender as idiossincrasias de cada tipo de domínio [Codenie et al. 1997]. O processo de análise de domínio identifica requisitos essenciais do contexto da aplicação baseado em experiências previamente publicadas, sistemas de software similares, experiências pessoais, e padrões e normas comuns. Durante a análise de domínio, os *hot spots* e os *frozen spots* começam a ser descobertos [Markiewicz and de Lucena 2001].

No domínio de Sistemas de Informação (SI), por exemplo, Frameworks de Aplicação para Sistemas de Informação (FASI) deveriam prover características para li-

Mecanismo de Busca	Artigos Retornados	Artigos Selecionados
ACM	263	160
CiteSeerX	67	4
Engineering Village	306	44
IEEE	296	150
Web of Science	46	5

Table 1. Artigos encontrados e selecionados na revisão da literatura.

dar com operações CRUD (*Create*, *Read*, *Update*, *Delete* - Criar, Ler, Atualizar e Eliminar dados do usuário), regras e processos de negócio, interação com o usuário baseadas em formulários, interação com software de persistência e representação de entidades de negócio. FASI geralmente proveem soluções para alguns desses aspectos de modo isolado, e o usuário é responsável por integrar as partes para construir a aplicação de software.

2.3. Combinação de FA e ESDM

Para compreender o estado da arte sobre a combinação dos conceitos de FA e DSDM, foi conduzida uma revisão exploratória da literatura seguindo os princípios de Revisão Sistemática [Kitchenham et al. 2009], usando as seguintes fontes de dados: ACM, CiteSeer, Engineering Village, IEEE, e Web of Science.

A revisão foi conduzida entre maio de 2010 e dezembro de 2011 e considerou artigos escritos em inglês ou português. A Tabela 1 mostra o número de artigos retornados em cada fonte de dados, e o número de artigos selecionados. É importante notar que o número de artigos retornados inclui duplicatas, isto é, o mesmo artigo foi retornado em diferentes mecanismos de busca.

A seleção foi baseada na pertinência do artigo ao assunto da revisão, depois de ler o título, resumo e introdução do artigo. Para cada artigo selecionado, foram analisadas as seguintes questões:

- Quais *hot spots* and *frozen spots* foram descritos nos artigos?
- Que padrões arquiteturais e decisões arquiteturais são discutidas?
- De que modo os conceitos de ESDM se relacionam com os conceitos de FA?

A principal conclusão foi que, embora a combinação de FA e ESDM pareça ser uma boa ideia, não há descrições de como implementar essa ideia. Poucos trabalhos discutem essa combinação e eles não descrevem questões básicas de implementação, tais como:

- o critério para definição de *hot spots* e *frozen spots* de um FA dirigido por modelos;
- os mecanismos de Inversão de Controle considerando que parte do código é automaticamente gerado;
- a representação de variabilidades e similaridades específicas de domínio;
- os paradigmas arquiteturais e padrões de projeto que podem ser usados para implementar o sistema.

Detalhes sobre a arquitetura da primeira versão do framework podem ser obtidos em [Almeida et al. 2009, da Silva 2010, Boff 2010]. A próxima seção descreve detalhes da segunda versão da arquitetura.

3. Arquitetura e Projeto da Nova Versão do Framework

O framework foi construído para realizar a geração e manutenção automática de componentes de software de SI com base em um metamodelo conceitual de domínio [Almeida et al. 2009]. Esta seção apresenta detalhes da macro-arquitetura do framework e suas especificidades notáveis.

3.1. Visão Geral da Arquitetura

Na primeira versão do framework, um único metamodelo foi usado para conceber todas as partes do SI. Transformações automáticas foram projetadas para gerar, de modo estático e não customizável, interfaces de usuário (IU), regras de negócio (RN) e *scripts* de banco de dados (BD) do SI. Havia considerável quantidade de replicação no código do framework, além de código relacionado às transformações espalhado e emaranhado no código da arquitetura.

Logo, o primeiro passo na refatoração desse framework foi criar novos metamodelos para expressão das diversas facetas de um SI. O metamodelo de domínio original, que era único, foi revisado e estendido, e três novos metamodelos foram derivados: um para descrever aspectos de Interação [da Costa et al. 2010], outro para descrever aspectos do Domínio de Negócio e outro para descrever Processos de Negócio [Loja et al. 2010]. Novas necessidades levaram a novas versões dos metamodelos de Domínio de Negócio, que pode ser encontrado em [Graciano Neto 2012], e de Interação com o Usuário [da Costa 2011].

Estes metamodelos são representados em XML pelas facilidades e independência de tecnologia oferecidas por esta notação. Na versão anterior, as definições de dados do metamodelo eram atreladas ao código do framework, o que reduzia o grau de manutenibilidade do framework e demandava mudanças de maior impacto em todas as partes do código.

A nova arquitetura é estruturada em componentes e é apresentada na Figura 1 sob o ponto de vista de *hot spots* e *frozen spots*. Aqueles que possuem a letra H depois do nome são *hot spots* enquanto que aqueles que possuem a letra F são *frozen spots*.

Para construir um SI utilizando o framework descrito é necessário instanciar, além das regras de transformação, cada um dos *hot spots*.

O componente de Apresentação/Interação usa o conceito de Estereótipo de Interface descrito em [da Costa et al. 2010] como um *hot spot*. Esse conceito consiste em uma abstração de intenção de interface que é independente da aplicação subjacente ou SI. Constitui uma representação genérica, extensível, abstrata e completável de uma interface como CRUD, Portal, e outras, o que permite classificá-lo como um *hot spot*. Como trata-se de uma abstração que deve ser instanciada e concretizada, por decisão de projeto, este conceito foi ofertado como um *hot spot* do domínio de SI para as aplicações geradas utilizando o FA descrito.

Essa definição ou essas definições de Estereótipos de Interface serão referenciadas pelas regras de transformação e servem de insumo e entrada para o transformador bem como o modelo de domínio de entrada. Logo, o modelo instanciado do *hot spot* tornar-se-á um modelo em alto nível de abstração de entrada para o transformador do

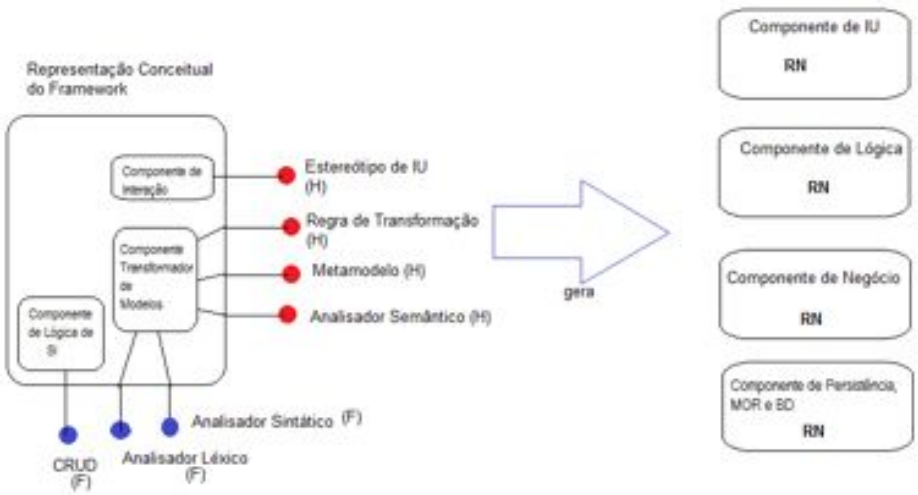

Figure 1. *Hot e frozen spots* do Framework.

framework. O modelo de domínio também é insumo dessa transformação para gerar o software completo.

A Figura 2 mostra o funcionamento geral do framework para geração automática de um SI com a utilização explícita de *hot spots*. O Componente de Interação com o Usuário (IU) oferta um *hot spot* de esteriótipo de interface (em suma, um modelo abstrato de interface). O Componente Transformador oferta regras de transformação como *hot spots*. Instancia-se tanto as regras de transformação, quanto o modelo de IU quanto outros modelos de origem (como Domínio e Processo de Negócio). Ao entregar todos estes modelos concretos como entrada para o transformador, ele responsabiliza-se por gerar o SI.

O componente de Lógica de SI guarda as definições genéricas comuns a lógica de SI. Através da análise de domínio verificou-se que CRUD é uma funcionalidade padrão ofertada por todo e qualquer SI. Logo, deve ser ofertada como *frozen spot*. Outras funcionalidades da lógica devem ser geradas manualmente.

O componente Transformador de Modelos guarda as definições necessárias para efetuar a transformação de modelos. Nele, identificou-se como *hot spot* o analisador semântico, regras de transformação e metamodelo. Analisadores léxico e sintático para os modelos de entrada são identificados como *frozen spots*.

Em relação ao software gerado, essa estrutura gera um produto de software com componentes coesos que interagem para gerar um SI operacional. Regras de negócio são replicadas nesses componentes para permitir que validações de dados sejam feitas em cada um dos componentes. Isto permite que cada componente possa ser utilizado por um usuário do framework, inclusive, de modo isolado, aderindo à definição integral de componente e reforçando a definição de framework de aplicação (um conjunto de componentes). No componente de negócio, são geradas as representações do negócio em software a partir do modelo de domínio de negócio.

A representação do metamodelo de domínio do framework é também identificada

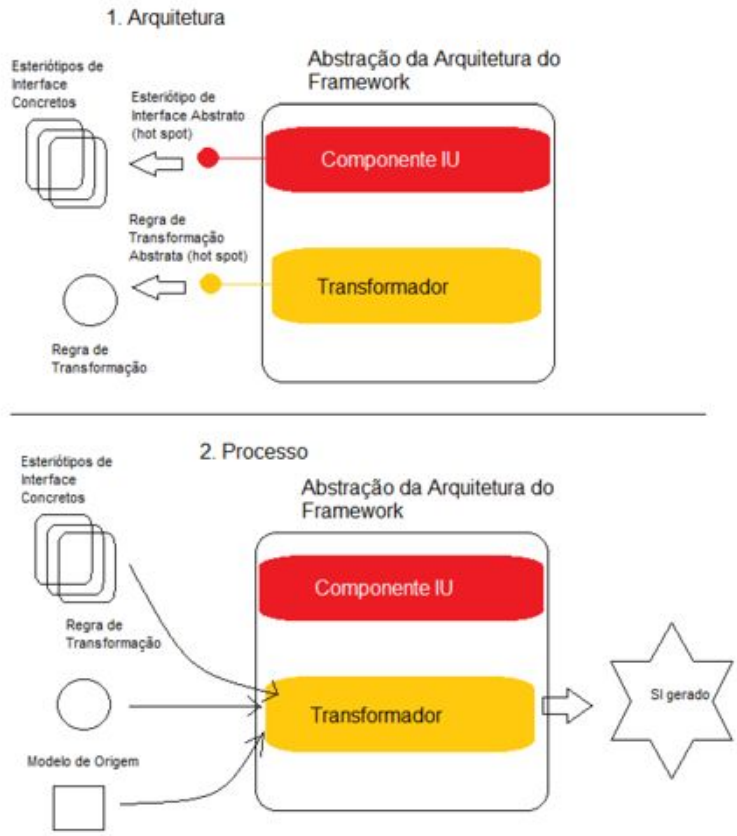

Figure 2. Processo simplificado de Instanciação do Framework.

como ponto de extensão (*hot spots*). Apesar de não ser convencional a relação "*é uma instância de*" para relacionar entidades de negócio com suas respectivas meta-entidades (ou o modelo e seu respectivo metamodelo), para o framework descrito essa relação é aceitável, uma vez que é o ato de completar os dados do metamodelo XML em nível conceitual (instanciar o metamodelo) que gera o modelo concreto. Então, a definição do metamodelo é identificada como um *hot spot*.

Em relação à transformação de modelos, pode-se ver o mapeamento de modelos previsto em ESDM como um processo de compilação que transforma um modelo fonte (escrito segundo o metamodelo fonte) em um modelo alvo (escrito segundo outro meta-modelo). Por isso, existem *hot spots* e *frozen spots* que possuem responsabilidades dentro do processo de compilação.

Assim, o processo de transformação (ou mapeamento de modelos) expresso como um processo de compilação convencional envolve análises léxica, sintática e semântica, e geração de código.

O framework recebe como entrada o metamodelo e o modelo em XML. Então, o framework checa se o modelo é condizente com o metamodelo.

A verificação é genérica e compreende a verificação da corretude das *tags* do

modelo em relação às *tags* especificadas no metamodelo (análise léxica). Além disso, o validador analisa se as *tags* que são internas umas às outras estão no lugar devido (na profundidade devida dentro da árvore do XML - análise sintática). E, por fim, esse validador verifica as restrições conceituais não explícitas no XML (análise semântica). Ou seja, o validador usa serviços dos componentes ImportaModelo, ImportaMetamodelo, AnalisadorLéxico, AnalisadorSintático e AnalisadorSemântico.

Seguindo o processo clássico de compilação, a transformação de modelos inicia com a análise léxica, passa pela análise sintática e segue para a análise semântica. Se as análises léxica, sintática e semântica são bem-sucedidas, então a geração de código é iniciada. Como o modelo fonte é validado sob as perspectivas léxica, sintática e semântica, a geração de código (transformação do modelo origem no modelo alvo) deve funcionar, salvo situações excepcionais, tais como problemas com permissões para escrever a saída em um diretório.

O Analisador Semântico é um *hot spot* no sentido que mudanças no metamodelo causam muitas modificações dentro dele. Não se trata, portanto, de um *hot spot* que pode ser estendido. Na verdade, são necessárias manipulações no código do transformador. No entanto, há uma identificação explícita de um ponto de mudança e um *hot spot* convencional poderia ser oferecido, permitindo que o usuário possa configurar regras de validação semânticas que devem ser especificadas sobre o metamodelo de origem, ativando essas regras no momento correto. Então, o Analisador Semântico do transformador de modelos pode ser considerado um *hot spot* em potencial.

Como foi escolhida a linguagem XML para representar modelos e metamodelos, o framework é independente do metamodelo. Isto é, se o metamodelo evolui, o framework não necessita ser alterados. Detalhes sobre a implementação das regras de mapeamento serão mostrados num artigo futuro.

Com relação a Regras de Negócio (RN), elas deveriam estar presentes nas classes de negócio e em todo componente gerado pelo transformador, isto é, checagens de consistência são feitas em todo componente. Uma vez que cada parte será gerada, não haverá problemas com manutenibilidade e redundância. Com essa decisão, adere-se mais completamente a uma das definições de framework de aplicação que diz que um FA é um conjunto de componentes coesos.

No tocante ao modo de instanciação do framework deste projeto, percebeu-se duas situações de instanciação não descritas na literatura. A primeira é a extensão de um framework através da oferta de um metamodelo como um *hot spot*. Nesse caso, para instanciar o framework, que caracteriza-se como uma aplicação semi-completa, e torná-lo uma aplicação completa, o que se faz é obter um modelo a partir de um metamodelo, tornando possível a iniciação da operação do transformador de modelos, transformando-o em uma aplicação efetivamente funcional.

Assim, um sub-produto da pesquisa desenvolvida é a conclusão de que, para frameworks de aplicação dirigidos por modelos, a instanciação de um metamodelo (que nesse caso é feita com o preenchimento de um documento XML) é um modo de instanciar frameworks não documentado na literatura.

Uma ideia próxima a essa abordagem é a dos frameworks estruturais, como JPA e Hibernate, que utilizam-se de documentos XML para efetuar parte da configuração do

mapeamento objeto-relacional. Entretanto, o documento XML da abordagem dos frameworks estruturais não possui característica de metamodelo, o que difere da abordagem realizada neste trabalho.

A outra situação acontece devido à necessidade de alteração do código do framework por intervenção direta e manual. Percebeu-se que para criar novas regras de transformação, não basta herdar da regra abstrata ofertada, mas também é necessário alterar o código do transformador para efetuar a chamada às novas regras criadas.

Essa modalidade de extensão do framework é complementar às outras abordagens, uma vez que surge a demanda de modificação do código em virtude da utilização da abordagem de Herança. Porém, ainda assim, é algo importante de se mencionar como efeito colateral da instanciação e uma realidade do desenvolvimento de software baseado em frameworks de aplicação.

Outro ponto que é gerado pelo framework é a camada de Mapeamento Objeto-Relacional (MOR), Persistência e Banco de Dados. Isso é necessário porque intenciona-se, futuramente, gerar um software de SI capaz de interagir com vários tipos de bancos de dados e SGBD que serão escolhidos pelo usuário do framework. Logo, haverá em essência Mapeamentos Objeto-Paradigma (MOP), em que *P* representa qualquer paradigma de armazenamento de dados, incluindo o banco de dados relacional, orientado a objetos e baseado em grafos ou arquivos convencionais. A ideia é especificar regras abstratas e a partir dessas originar classes que representam regras de transformação para geração de código de MOR e persistência.

O SI gerado pelo framework também goza da independência de Sistema Operacional (SO), uma vez que a tecnologia-alvo do SI gerado pelo framework é Java. Tal tecnologia baseia-se em *bytecodes* e máquinas virtuais, o que torna o SI gerado também independente de SO. Uma ferramenta de apoio à prototipação e apoio à instanciação do framework também foi implementada como prova de conceito da pesquisa realizada. Não será detalhada aqui por questões de espaço. A próxima seção apresenta as considerações finais e os trabalhos futuros.

4. Considerações Finais

Este artigo apresentou os resultados obtidos a partir de uma dissertação de mestrado. Apresentou uma solução arquitetural para um Framework de Aplicação (FA) para Sistemas de Informação (SI) que usa a abordagem de Engenharia de Software Dirigido por Modelos (ESDM) para síntese de software. Ela representa uma evolução de uma versão anterior de um FA de mesma natureza que começou a ser desenvolvido em 2005[Almeida et al. 2009], tornando a arquitetura mais flexível às mudanças naturais do domínio de SI.

O uso de tais conceitos torna a produção de software de SI menos propensa a erros, uma vez que a produção de código é realizada a partir de transformação de modelos. Isso reduz o tempo e o custo de manutenção de software, que representa percentual considerável de seu custo de produção. A nova arquitetura evita redundância de código por explorar os conceitos de reúso e herança da orientação a objetos para estabelecer uma hierarquia de regras de transformação.

Uma revisão exploratória com base nos princípios de Revisão Sistemática foi con-

duzida e uma conclusão importante desta revisão é que há escassez de literatura sobre FA para construção de SI baseados em ESDM. E, dentre os trabalhos encontrados, a maioria utiliza o termo *framework* de modo indiscriminado para denotar qualquer aparato de software que apóie o processo de desenvolvimento, não configurando realmente frameworks de aplicação. Além disso, os trabalhos não discutem ou explicitam *hot spots* e*frozen spots* daqueles frameworks como este trabalho apresente.

Dentre as contribuições diretas deste trabalho, podem-se citar, portanto:

1. Uma solução arquitetural que identifica explicitamente *hot spots* e *frozen spots* de um FA para SI com ESDM, algo não encontrado na literatura;
2. Aderência da macro-arquitetura apresentada à definição original de FA, composta por *hot spots*, *frozen spots*, e *inversão de controle* (uma vez que durante a revisão de literatura, constatou-se a existência de vários trabalhos tratando de frameworks sem referir-se a esses termos inerentes ao assunto);
3. A disponibilização do processo de instanciação desta categoria de FA, algo também não encontrado na literatura;
4. Uma solução arquitetural de FA para SI que gera um software integrado e coeso, contemplando os principais aspectos de um SI, não ofertando apenas uma destas características (como Interação, Domínio ou Regras e Processos de Negócio) de modo isolado.

Em relação a trabalhos correlatos, o Eclipse Modeling Framework (EMF) [1], associado ao ambiente de desenvolvimento Eclipse, tem sido usado para auxiliar no processo de geração de código de abordagens dirigidas por modelos, tais como [Marzullo et al. 2008]. Entretanto, os *hot spots* dessa ferramenta não são explícitos e um dos possíveis *hot spots* (as regras de transformação do framework) é laborioso para se estender e gerar novas regras. Além disso, a tecnologia de geração de código disponível requer que as regras sejam especificadas usando uma linguagem para especificar regras de transformação independentes de plataforma como ATL e QVT [Jouault 2008], o que demanda tempo. Assim, optou-se por não usar tecnologias como o EMF devido às suas limitações e implementar um framework customizado com princípios de ESDM.

Na revisão de literatura conduzida, apenas três estudos foram encontrados descrevendo especificamente a modelagem e geração de SI usando um FA de modo semi-automático ou baseado em modelos [Okanović et al. 2010], [Langegger et al. 2006], e [Almeida et al. 2009]. Os outros exemplares eram FA apenas para ESDM ou para SI, não sendo diretamente correlatos.

Dois desses estudos [Okanović et al. 2010, Almeida et al. 2009] não apresentam a arquitetura dos frameworks, as características de Inversão de Controle (inerente a frameworks de aplicação [Fowler 2005]) e a identificação de *hot/frozen spots*. Langegger et. al [Langegger et al. 2006] mencionam o conceito de *spots de Interação* (pontos de extensão relacionados à interação com o usuário), ilustrando também uma proposta arquitetural em alto nível e indiretamente indicando a existência de Inversão de Controle. Entretanto, eles não discutem detalhes sobre outros *hot spots* ou *frozen spots* do FA proposto.

Existe um protótipo da nova versão do framework operacional com ideias implementadas para validação dos pontos discutidos. O trabalho está sendo estendido e

[1] http://www.eclipse.org/modeling/emf/

desmembrado para sub-projetos que darão continuidade às características projetadas. Alguns componentes já estão implementados e estão sendo integrados, como o componente analisador semântico e o componente de processo de negócio [de Oliveira et al. 2011]. Ele já possui mais de 5000 mil linhas de código.

Trabalhos futuros incluem uma validação experimental da nova versão do FA, a investigação de conjuntos mínimos de modelos para geração de software em outros domínios, como *middleware*, compiladores, segurança, dentre outros; a representação, também em XML, dos metamodelos de processo de IU, associando os conceitos fundamentais de um SI para gerar um software de SI completo, integrado e funcional; a mensuração dos ganhos obtidos com este FA em relação à versão anterior dele e a outros frameworks para SI comerciais.

References

Almeida, A. C., Boff, G., and Oliveira, J. L. (2009). A Framework for Modeling, Building and Maintaining Enterprise Information Systems Software. In *Anais do XXIII Simpósio Brasileiro de Engenharia de Software*, pages 115–125. Fortaleza, Brasil.

Boff, G. (2010). Arquitetura e implementação de mecanismos para suporte a regras de negócio em sistemas de informação. Master's thesis, Instituto de Informática - Universidade Federal de Goiás.

Butler, M. J., Petre, L., and Sere, K., editors (2002). *Model Driven Engineering*, volume 2335 of *Lecture Notes in Computer Science*. Springer.

Codenie, W., De Hondt, K., Steyaert, P., and Vercammen, A. (1997). From custom applications to domain-specific frameworks. *Communications of ACM*, 40:70–77.

da Costa, S. L. (2011). Uma abordagem baseada em modelos para construção automática de interfaces de usuário para sistemas de informação. Master's thesis, Instituto de Informática - Universidade Federal de Goiás.

da Costa, S. L., Graciano Neto, V. V., Loja, L. F. B., and de Oliveira, J. L. (2010). A metamodel for automatic generation of enterprise information systems. In *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 45–52. UFBA. Salvador, BA, Brasil.

da Silva, W. C. (2010). Gerência de interfaces para sistemas de informação: Uma abordagem baseada em modelos. Master's thesis, Instituto de Informatica - Universidade Federal de Goias.

de Almeida, A. C. (2010). Um componente para geração e evolução de esquemas de bancos de dados como suporte à construção de sistemas de informação. Master's thesis, Instituto de Informática - Universidade Federal de Goiás.

de Oliveira, J. L., Loja, L. F. B., da Costa, S. L., and Neto, V. V. G. (2011). Um componente para gerência de processos de negócio em sistemas de informação. In *Anais do VII Simpósio Brasileiro de Sistemas de Informação*, pages 250 – 261.

Fayad, M. and Schmidt, D. C. (1997). Object-oriented application frameworks. *Communications of ACM*, 40(10):32–38.

Fowler, M. (2005). Inversion of control. http://martinfowler.com/bliki/InversionOfControl.html.

Graciano Neto, V. V. (2012). Evolução de uma arquitetura para frameworks de aplicação de sistemas de informação - uma abordagem de desenvolvimento dirigido por modelos. Master's thesis, Instituto de Informática - Universidade Federal de Goiás.

Graciano Neto, V. V., da Costa, S. L., and Oliveira, J. L. (2010). Lições Aprendidas sobre Desenvolvimento Dirigido por Modelos a partir da refatoração de uma ferramenta. In *Anais do Encontro Anual de Computação (ENACOMP)*, pages 68–75. Catalão, Brasil.

Jouault, F. (2008). ATL: A model transformation tool. *Science of Computer Programming*, 72(1):31 – 39. Special Issue on Second issue of experimental software and toolkits.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering - a systematic literature review. *Information Software Technology*, 51(1):7–15.

Kleppe, A. G., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Langegger, A., Palkoska, J., and Wagner, R. (2006). Davinci - a model-driven web engineering framework. *International Journal of Web Information Systems*, 2:119 – 134.

Loja, L. F. B., Graciano Neto, V. V., da Costa, S. L., and de Oliveira, J. L. (2010). A business process metamodel for enterprise information systems automatic generation. In *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 37–44, Salvador, BA, Brasil. UFBA.

Mailloux, M. (2010). Application frameworks: how they become your enemy. In *Proceedings of the ACM International Conf. on OOP Systems Languages and Applications*, SPLASH '10, pages 115–122, New York, NY, USA. ACM.

Markiewicz, M. E. and de Lucena, C. J. P. (2001). Object oriented framework development. *Crossroads*, 7:3–9.

Marzullo, F. P., de Souza, J. M., and Blaschek, J. R. (2008). A Domain-Driven Development Approach for Enterprise Applications, Using MDA, SOA and Web Services. In *10th IEEE International Conference on E-Commerce Technology (CEC 2008) / 5th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008), July 21-14, 2008, Washington, DC, USA*, pages 432–437.

Okanović, V., Donko, D., and Mateljan, T. (2010). Frameworks for model-driven development of web applications. In *Proc. of 9th WSEAS International Conference on Data Networks, Communications, Computers*, DNCOCO'10, pages 67–72, Stevens Point, Wisconsin, USA. WSEAS.

Santos, A. L., Lopes, A., and Koskimies, K. Modularizing framework hot spots using aspects. In *Proceedings of XI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2006)*, Sitges, Barcelona, Espanha.