Reflexões sobre Síntese Automática de Sistemas de Informação Temporais através de Modelos

Valdemar Vicente Graciano Neto

[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

vgracianoneto@gmail.com

Resumo. *Software de Sistemas de Informação (SI) naturalmente precisam armazenar informações temporais. Para tanto, é necessário que o banco de dados seja do tipo temporal. Entretanto, os sistemas gerenciadores de banco de dados modernos dão suporte a apenas poucos tipos de aspectos temporais. Em virtude disso, é comumente necessário um componente de software que administre a temporalidade do software de SI e a temporalidade do próprio banco de dados. Este artigo apresenta reflexões e decisões de projeto que servirão de insumo para conceber uma arquitetura dirigida por modelos para síntese automática de SI com suporte a temporalidade.*

1. Introdução

O desenvolvimento de software de Sistemas de Informação (SI) mantém-se ainda 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 orçamentos e prazos reduzidos.

Uma complexidade notável com a qual é preciso lidar é o tratamento de aspectos temporais em SI. Ela surge naturalmente em várias categorias de software de SI e consiste na necessidade de manter informações que são alteradas ao longo do tempo.

SI são naturalmente associados a Bancos de Dados (BD) e os Sistemas Gerenciadores de Banco de Dados (SGBD) modernos oferecem poucos recursos para tratamento de temporalidade. Georg concluiu, a partir de um estudo conduzido sobre temporalidade nos SGBD Objeto-Relacionais Oracle e PostgreSQL, que, apesar de não impedir a aplicação de temporalidade em BD, essas tecnologias não dão o suporte necessário para fazê-lo de forma transparente e eficiente [de Castro Georg 2010].

O Desenvolvimento de Software Dirigido por Modelos (DSDM) [Stahl et al. 2006] é uma forma de lidar com as complexidades do processo de desenvolvimento de SI ao substituir o código-fonte por modelos abstratos que representam especificações em alto-nível de abstração.

O código é gerado automaticamente a partir dos modelos de modo que o código não demande manipulação direta, atividade onerosa e propensa a erros.

Este artigo apresenta resultados preliminares, reflexões e decisões de projeto que servirão de insumo para conceber uma arquitetura dirigida por modelos para geração automática de aspectos de gerência de temporalidade em projetos de desenvolvimento de software de SI. Para tratar deste tema, a Seção 2 discute o embasamento teórico desta

pesquisa; a Seção 3 discute decisões de projeto para arquitetura de uma ferramenta de transformação de modelos para síntese de SI temporal; e a Seção 4 apresenta as conclusões e trabalhos futuros.

2. Embasamento Teórico

Esta seção apresenta os pilares teóricos sobre os quais esta pesquisa se apoia.

2.1. Temporalidade em SI e BD

Sistemas de Informação (SI) de grande porte são inerentemente temporais. Isto é, necessitam tratar a evolução sofrida pelos dados e guardar o histórico de modificações ocorridas.

Como as informações de um SI são comumente gravadas em um Banco de Dados, é necessário prover modos de tratar a temporalidade em nível de aplicação, uma vez que os Sistemas Gerenciadores de Banco de Dados (SBGD) modernos não dão vasto suporte ao tratamento de temporalidade [de Castro Georg 2010].

Bancos de Dados (BD) convencionais representam objetos do mundo real em um único momento do tempo, o atual. BD temporais, por sua vez, permitem que se pesquise por estados anteriores (passado) ou posteriores (futuro), utilizando uma linguagem de consulta temporal. Por estado futuro deve-se entender que a informação já está presente no banco de dados mas somente será válida a partir de um instante futuro [de Castro Georg 2010], como quando um débito em uma conta ocorre num final de semana, mas só sera efetivado no primeiro dia útil seguinte.

Para tanto, porções de código são acrescidas à arquitetura do SI bem como ao código de manipulações de dados do BD (*stored procedures*) para gerenciar a temporalidade do SI.

Um modelo de dados convencional de BD apresenta apenas duas dimensões: registro (tupla) e campo. Neste caso, a mudança de qualquer valor de instância implica na perda do valor anterior do campo no qual o dado foi modificado.

Os modelos temporais acrescentam uma terceira dimensão: o tempo. Esta dimensão atribui uma informação temporal a cada valor. Desta forma, a mudança de qualquer valor não inutiliza o valor anterior.

Em aplicações de BD utiliza-se diversos tipos de dados temporais com diferentes graus de precisão. Para tanto, em BD temporais estabelece-se os conceitos de *tempo de validade* e *tempo de transação*.

Tempo válido (ou de validade) é o tempo em que o valor associado a ele é válido no mundo real. É um valor que é explicitamente informado pelo usuário. Ou seja, o usuário tem total controle sobre o tempo de validade de uma informação. Isso demanda o acréscimo de dois campos a cada tabela: TVI e TVF que representam, respectivamente, o tempo válido inicial e o tempo válido final.

Tempo de transação indica o instante em que os valores foram efetivados no BDT. A informação associada ao tempo de transação será válida (atual no banco de dados) enquanto não for excluída logicamente. Este tipo de informação temporal é inacessível ao usuário, pois é gerada e gerenciada pelo SGBD. Consequentemente, o tempo de transação

não pode ser posterior ao tempo atual do sistema. Um exemplo de controle de tempo de transação é o uso do tipo de dados TIMESTAMP em logs de operações do SGBD.

Tanto em BD de tempo de transação quanto em BD de tempo válido, uma alteração gera exatamente uma nova tupla.

Logo, em virtude de uma quantidade elevada de tuplas em uma mesma tabela e devido ao fato de que nem todas elas representam dados válidos no momento presente, é necessário filtrar os dados todas as vezes que else são disponibilizados para o usuário. Assim, é necessário gerar funcionalidades de software que gerenciem tanto a alteração dos dados no banco de dados de modo a possibilitar a atualização dos dados temporais quanto a leitura dos dados em virtude de consultas realizadas a estes tipos especiais de BD.

Em virtude da necessidade de gerar código manualmente e acrescentá-lo na arquitetura do SI (algo repetitivo e propenso a erros), vislumbrou-se a aplicação de princípios de Desenvolvimento de Software Dirigido por Modelos (DSDM) para a geração destas porções de software a partir de modelos em alto-nível de abstração, tema discutido na próxima seção.

2.2. Desenvolvimento de Software Dirigido por Modelos

Existem modelos específicos para cada fase do processo de desenvolvimento de software. O modelo de requisitos é usado como entrada para discussão e produção de modelos de projeto e arquitetura. Estes modelos são levados em consideração para a estruturação do código-fonte e o código-fonte é o insumo para especificação de casos de teste. O processo de desenvolvimento de software resume-se a uma série de *transformações entre modelos* conduzidas majoritariamente de modo manual [Graciano Neto et al. 2010].

DSDM quebra esse paradigma por automatizar este processo de transformação. O objetivo deste novo modelo prescritivo de processo de desenvolvimento de software é elevar o nível de abstração na produção de software trazendo ganhos como geração automática de código, produtividade, manutenibilidade, portabilidade, reusabilidade e rastreabilidade [Kleppe et al. 2003, Miller and Mukerji 2003, Graciano Neto et al. 2010].

Para tanto é necessário um software transformador de modelos e uma definição de transformação - um conjunto de regras que especificam como um modelo de origem pode ser transformado em um modelo destino (código-fonte) [Kleppe et al. 2003]. Estas regras ou definições de transformação devem ser plugáveis à ferramenta de transformação.

Para tornar a automação uma realidade, os modelos precisam ter um significado bem definido [Mellor et al. 2003]. Uma forma de prover tal significado é criar *metamodelos*, ou seja, modelos que fazem asserções sobre o que pode ser expresso em outros modelos [Seidewitz 2003]. Além disso, eles proveem sintaxe e semântica definidas para os modelos produzidos a partir deles.

A especificação MDA (*Model-Driven Architect*) - especificação seminal do OMG para guiar a produção de software segundo DSDM - não apresenta propostas arquiteturais para as ferramentas de transformação dirigidas por modelos. Logo, cada iniciativa implementa a transformação de uma forma diferente.

A MDA propõe que um *Modelo Independente de Plataforma (Plataform Inde-*

pendent Model - PIM) - como um modelo de domínio - seja usado como entrada para uma ferramenta de transformação para produzir automaticamente um *Modelo Específico de Plataforma (Plataform Specific Model - PSM)* - como um modelo arquitetural ou de projeto, ou código-fonte.

A próxima seção mostra como estas ideias se relacionam para estabelecer uma arquitetura de uma ferramenta capaz de sintetizar software de SI com suporte à temporalidade.

3. Decisões de Projeto para a Ferramenta de Síntese de SI

Os modelos utilizados para gerar software de SI representam, em geral, os três principais aspectos de um SI: domínio, interação e processo de negócio [da Costa et al. 2010, Loja et al. 2010, de Oliveira et al. 2011]. O domínio, entretanto, permite gerar um SI completo uma vez que, a partir dele, é possível gerar o BD, os objetos de negócio, as funcionalidades CRUD (Criar, Ler, Atualizar e Deletar) de persistência, e as telas para interação [Almeida et al. 2009].

Logo, pela suficiência do modelo de domínio como modelo de origem, por decisão de projeto, optou-se por utilizar apenas este modelo como modelo de entrada para a criação de um SI completo temporal. Da Costa et al. apresentaram um metamodelo para geração automática de Sistemas de Informação Empresariais [da Costa et al. 2010]. Este metamodelo foi ampliado e estendido para acrescentar características de domínio que são necessárias para a geração de um SI completo. Esta extensão é apresentada em [Graciano Neto 2012] mas não é reproduzida aqui por questões de espaço.

Para a síntese de um SI com características temporais através de um processo de DSDM, foram tomadas as seguintes decisões de projeto:

1. **O SI temporal (e consequentemente o BD) gerado será dotado de tempo de validade e tempo de transação**;
2. **O padrão arquitetural adotado para o SI temporal gerado é o de Camadas** [Fowler 2002];
3. **Todos os elementos do metamodelo de origem (metadados e respectivos atributos) serão dotados de tempo inicial e final de validade e transação.** Isso implica na adição de quatro campos às tabelas geradas a partir de cada um destes metadados;
4. **O metamodelo de domínio (origem) a partir do qual deriva-se o modelo de domínio de origem é especificado em XML**. [Graciano Neto 2012] mostra uma versão XML completa do metamodelo. Um trecho deste metamodelo instanciado é mostrado à frente;
5. **O paradigma de geração do SI temporal é o *preenchimento de template de código*** [Wittman 2010]. Isto é, a partir da definição XML do modelo de entrada, são extraídas as informações para preencher templates de código que integrarão o SI temporal final. Os *scripts* de geração do BD temporal, as funcionalidades de gerência de temporalidade, e o CRUD da aplicação são gerados através do preenchimento de *templates* de código. A geração da interface é feita sob o paradigma IF-ELSE [Ma and Yang 2010, Graciano Neto and de Oliveira 2011], isto é, para cada tipo de dado de entrada, um tipo específico de elemento gráfico de interação é gerado (rótulo, campo de texto, etc.);

6. **A temporalidade é inserida na ferramenta de transformação, não no metamodelo de origem**. Temporalidade em SI não é um assunto de domínio de grande parte dos engenheiros de software. Além disso, seriam necessários trechos de código referentes a temporalidade tanto no metamodelo quanto na ferramenta de transformação de modelos. Para evitar problemas de manutenção e evolução da ferramenta de transformação de modelos, optou-se por manter as questões de geração de temporalidade apenas na ferramenta de transformação;
7. **A geração de código dirigida por modelos será adotada em grande parte do processo**. Na geração das funcionalidades CRUD da camada de aplicação, a geração de código será realizada de modo convencional por ser uma funcionalidade padrão para qualquer tipo de SI e não exige informações específicas do modelo de origem;
8. **O padrão arquitetural para trânsito de dados dentro da arquitetura do SI gerado é o padrão de *Transfer Object*** [Fowler 2002];
9. **A geração de BD temporais inicialmente será voltada ao SGBD Postgres**;
10. ***Stored procedures* serão geradas para realizar o CRUD temporal dentro do BD**. Isto é, as atividades básicas de um banco de dados como Criar, Ler, Atualizar e Deletar valores serão invocados pela camada de Persistência uma vez que são necessárias manipulações nas tuplas de cada tabela;

As decisões de projeto 2, 8 e 9 são herdadas de uma versão anterior de um framework dirigido por modelos para geração de SI [Almeida et al. 2009] sobre o qual esta ferramenta se baseia.

Para ilustrar o processo de geração do SI temporal, o Código 1 traz um trecho de entrada do metamodelo em XML com a especificação de uma Entidade Forte chamada Pessoa Física que possui dois atributos alfanuméricos: cpf e nome. Dentro da tag *entidade* consta o nome da entidade em questão, um valor booleano que representa se a entidade é forte ou fraca, e o mnemônico desta entidade, que trata-se de um identificador alfanumérico único que é utilizado para dar nome à tabela gerada no banco de dados.

Esta entidade é um metadado de negócio. Ela possui dois atributos. Cada atributo possui um mnemônico (que dará nome ao campo da tabela gerada), cardinalidades mínima e máxima (referentes à quantidade de valores que aquele campo pode receber), um valor booleano que diz se aquele campo será parte de chave, e outro que diz se os valores daquele campo devem ser únicos. O atributo CPF possui um tamanho de 11 caracteres e não possui valor *default*. O campo 'nome' não é único e nem parte de chave, possui um tamanho de 50 caracteres e não possui valor *default*.

Code 1. Trecho do modelo de entrada em XML.

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <aplicacaoSI>
3  <metadadoNegocio>
4          <entidade>
5                  <nomeEntidade>Pessoa Física</nomeEntidade>
6                  <entidadeForte>true</entidadeForte>
7                  <mnemonico>PesFis</mnemonico>
8          </entidade>
9          <atributos>
10                 <atributo>
```

```
                    <mnemonico>cpf</mnemonico>
                    <cardinalidadeMaxima>1</cardinalidadeMaxima>
                    <cardinalidadeMinima>1</cardinalidadeMinima>
                    <ehParteChave>true</ehParteChave>
                    <ehUnico>true</ehUnico> <!-- TRUE ou FALSE -->
                    <!-- ALFANUMERICO -->
                    <tamanhoAlfanumerico>11</tamanhoAlfanumerico>
                    <valorDefaultAlfanumerico></valorDefaultAlfanumerico>
                  </atributo>
                  <atributo>
                    <atributo>
                    <mnemonico>nome</mnemonico>
                    <cardinalidadeMaxima>1</cardinalidadeMaxima>
                    <cardinalidadeMinima>1</cardinalidadeMinima>
                    <ehParteChave>false</ehParteChave>
                    <ehUnico>false</ehUnico> <!-- TRUE ou FALSE -->
                    <!-- ALFANUMERICO -->
                    <tamanhoAlfanumerico>50</tamanhoAlfanumerico>
                    <valorDefaultAlfanumerico></valorDefaultAlfanumerico>
                  </atributo>
        </atributos>
</metadadoNegocio>
</aplicacaoSI>
```

A partir destes dados é possível gerar o código SQL (Código 2) referente à criação do banco de dados com temporalidade, com a tabela de Pessoa Física e os respectivos atributos. A síntese da camada de gerência de temporalidade (que estaria acima da camada de persistência) bem como da aplicação, negócio e interação serão ocultadas por questões de espaço e devem ser apresentadas em um trabalho posterior.

Code 2. Trecho de código gerado a partir do trecho de metamodelo apresentado.

```
CREATE DATABASE Aplicacao;
CREATE TABLE PesFis(
          CPF varchar(11),
          Nome varchar(50),
          Dt_IniVal TIMESTAMP NOT NULL,
          Dt_FimVal date,
          Dt_IniTran TIMESTAMP NOT NULL,
          Dt_FimTran TIMESTAMP,
          PRIMARY KEY (CPF, Dt_IniVal, Dt_IniTran)
          );
```

Como pode-se notar, a tabela agora é temporal. Logo, qualquer alteração em qualquer registro gera uma nova linha na tabela com os dados atualizados. As datas de início de validade e transação precisam fazer parte da chave primária da tabela para permitir que uma pessoa inscrita sob um mesmo CPF possua diferentes registros nesta tabela, cada um correspondente a uma alteração de validade de um dado ou transação realizada sobre aquele dado.

4. Conclusão

Este artigo apresentou decisões de projeto que servem de insumo para a concepção de uma solução dirigida por modelos para a síntese automática de SI temporais.

Como trabalhos futuros destacam-se a possibilidade de expandir a viabilidade desta abordagem para outros SGBD que não apenas o SGBD Postgres; a geração de outras funcionalidades que não apenas as funcionalidades CRUD uma vez que SI modernos demandam outras funcionalidades como a emissão de relatórios, dentre outras; aumentar a flexibilidade de aparência nas telas geradas, uma vez que a utilização do paradigma IF-ELSE engessa as telas geradas nesta metodologia [da Costa 2011, Da Silva and de Oliveira 2009, da Costa and de Oliveira 2010]; tornar o SI gerado orientado a processos.

Além disso, pretende-se conduzir um estudo sobre o impacto da temporalidade no transformador de modelos. Ou seja, a temporalidade deveria ser modelada como um interesse transversal à especificação das regras de transformação, ou deveria ser modular? Ela deve ser especificada apenas no transformador ou o acréscimo ao metamodelo de origem traria algum benefício/malefício, como replicação, problemas de manutenção ou nenhum problema desta natureza?

Cabe ainda investigar, através de testes, a eficiência desta abordagem, isto é, quão grande é o ganho em tempo/recursos humanos ao utilizar esta abordagem ao invés das abordagens tradicionais de desenvolvimento de software.

Como contribuição direta deste trabalho destacam-se as decisões de projeto para a concepção de uma ferramenta dirigida por modelos que permite a síntese de um SI completo e funcional com características temporais, evidenciando as características arquiteturais da solução gerada pela ferramenta e um modo de realizar tal geração.

A implementação destas ideias já está em andamento e já apresenta resultados promissores. Isto evidencia a importância e eficácia da pesquisa e resultados obtidos até o momento, além de ampliar o acervo de projetos em que o Desenvolvimento de Software Dirigido por Modelos pode ser aplicado com sucesso.

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.

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. and de Oliveira, J. L. (2010). Construção e manutenção automática de interfaces com o usuário dirigidas por modelos para sistemas de informação. In *Anais do III Workshop de Teses e Dissertações em Sistemas de Informação do VI Simpósio Brasileiro de Sistemas de Informação*.

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. and de Oliveira, J. L. (2009). Gerência de interface homem-computador para sistemas de informação empresariais: uma abordagem baseada em modelos. *iSys - Revista Brasileira de Sistemas de Informação, Vol. 2, 2009*.

de Castro Georg, R. (2010). Bancos de dados temporais: Estudo sistematizado e aplicabilidade em sistemas gerenciadores de bancos de dados atuais. Trabalho de Conclusão de Curso - Instituto de Informática - UFG.

de Oliveira, J. L., Loja, L. F. B., da Costa, S. L., and Graciano Neto, V. V. (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.

Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

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.

Graciano Neto, V. V. and de Oliveira, J. L. (2011). An early aspect for model-driven transformers engineering. In *Proceedings of the 2011 international workshop on Early aspects*, EA '11, pages 7–11, Porto de Galinhas, PE, Brasil. ACM.

Kleppe, A., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture - Practice and Promise*.

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.

Ma, K. and Yang, B. (2010). A Hybrid Model Transformation Approach Based on J2EE Platform. volume 3, pages 161–164, Los Alamitos, CA, USA. IEEE Computer Society.

Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Guest editors' introduction: Model-driven development. *IEEE Software*, 20(5):14–18.

Miller, J. and Mukerji, J. (2003). MDA Guide Version 1.0.1. Technical report, Object Management Group (OMG).

Seidewitz, E. (2003). What models mean. *IEEE Software*, 20(5):26–32.

Stahl, T., Voelter, M., and Czarnecki, K. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons.

Wittman, P. (2010). MDA using AndroMDA. http://www.wittmannclan.de/ptr/cs/mdaandromda.pdf.