Compiladores para as Linguagens da Abordagem WED-flow

Eduardo D. Filho[1], **Bruno Padilha**[1], **João E. Ferreira**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 – 05508-090 – São Paulo – SP – Brasil

{edudf, brunopadilha, jef}@ime.usp.br

Abstract. *WED-flow (Work, Event and Data-flow) is a novel approach to model and implement Process-Aware Information Systems (PAIS) based on events, data and tasks in Business Processes. Instance processing is carried out in a transactional environment defined over a relational database. However, WED-flow definitions are expressed either in an abstract high-level set notation or in an intermediate declarative language named WED-SQL. Both languagem must undergo a translation process to SQL in order seamless integrate models and implementation in a relational database. This work proposes the specification of formal grammars and the implementation of compilers to translate both languages to the targeted SQL language.*

Resumo. *A abordagem WED-flow propõe modelar e implementar Sistemas de Informação Cientes de Processos (PAIS) a partir da especificação de eventos, dados e tarefas de um processo de negócio, oferecendo a execução de instâncias do processo em um ambiente transacional definido sobre um banco de dados. A implementação atual da WED-flow permite que as definições sejam especificadas por meio de uma notação de conjuntos ou por meio de uma linguagem intermediária chamada WED-SQL. Ambas as linguagens devem ser traduzidas para SQL para simplificar a integração com bancos de dados relacionais. Este trabalho propõe a especificação de gramáticas formais e a implementação de compiladores dessas duas linguagens para a linguagem alvo SQL.*

1. Introdução

Com os avanços tecnológicos na área de tecnologia da informação, organizações estão sujeitas à intensificação da necessidade de reagir rapidamente a mudanças nas regras de negócio e incorporação de novas regras para manter seus processos de negócio atualizados. Essa demanda por sistemas de informação flexíveis, que facilitam a aplicação de mudanças no processo de negócio, popularizou os chamados Sistemas de Informação Cientes de Processos (PAIS, do inglês *Process-Aware Information Systems*). PAIS são definidos como sistemas de *software* que gerenciam e executam processos operacionais envolvendo pessoas, aplicações e fontes de informação com base em modelos de processos [Dumas et al. 2005].

As abordagens formais como redes de Petri [Murata 1989] e álgebra de processos [Bergstra et al. 2001] realizam com sucesso o gerenciamento de processos de negócio (*Business Process Management*) oferecendo verificação formal do modelo. Essas abordagens são limitadas pois embora funcionem bem para processos estáticos especificados *a priori*, não apoiam mudanças incrementais nos processos. Outras abordagens propõem

o uso de linguagens como WSBPEL (*Web Services Business Process Execution Language*) [Jordan et al. 2007], que oferece flexibilidade para apoiar alterações incrementais em processos, mas o poder de especificação da linguagem WSBPEL apresenta desafios quanto à verificação formal e checagem de modelo [Ferreira et al. 2010]. Para apoiar o gerenciamento de processos de negócio dinâmicos e evolutivos de forma consistente foi proposta a abordagem WED-flow [Ferreira et al. 2012]. A abordagem WED-flow realiza o gerenciamento de processos de negócio de forma ciente de contexto, cuja importância é evidenciada em [vom Brocke et al. 2021].

A linguagem de modelagem WED-flow [Ferreira et al. 2012] é baseada na notação de conjuntos para especificar um processo. Em [Padilha et al. 2018], foi criada uma linguagem intermediária baseada em SQL chamada WED-SQL. A linguagem WED-SQL serve para definir as estruturas de um modelo em um nível mais baixo de abstração do que a linguagem de modelagem WED-flow e também para permitir o controle e monitoramento da execução de instâncias do modelo.

Os comandos WED-SQL devem ser traduzidos para SQL. A tarefa de identificar uma entrada do programa como um comando WED-SQL específico e de segmentar a entrada de modo que o programa seja capaz de atribuir valor sintático a cada uma das partes é uma tarefa complexa e sua implementação está sujeita a erros de naturezas diversas. O código do interpretador, por não seguir uma arquitetura formal de compiladores, era de difícil manutenção, tornando custosa a evolução das linguagens usadas na abordagem.

Para superar tal limitação, este trabalho apresenta a especificação de gramáticas formais e a implementação de compiladores dessas duas linguagens para a linguagem alvo SQL. Esses compiladores foram utilizados para a implementação do Sistema de Automatização de Autópsias (SISAUT) [Ferreira et al. 2017], conforme apresentado na Seção 5.

2. WED-flow

A modelagem de um sistema WED-flow é composta por estruturas que definem o modelo. As seguintes estruturas compõem uma modelagem WED-flow:

- WED-attributes: Representam os atributos do ambiente relacional que têm importância na especificação do *workflow*.
- WED-state: É uma configuração específica de valores para todos os WED-attributes definidos.
- WED-condition: Uma WED-condition é um predicado SQL escrito em função dos WED-attributes que representa uma condição de captura de eventos. Um WED-state satisfaz uma WED-condition caso sua configuração de valores de atributos faça com que o predicado seja verdadeiro.
- WED-transitions: São funções que alteram os WED-attributes levando a instância de um WED-state a outro, ou seja, são da forma $t : \mathcal{S} \rightarrow \mathcal{S}$.
- WED-triggers: Estruturas que associam uma WED-condition a uma WED-transition de forma que quando a WED-condition for satisfeita, a WED-transition associada será executada. Uma WED-trigger é denotada por uma tupla $g = \langle c, t \rangle$, em que c é uma WED-condition e t uma WED-transition.
- WED-flow: Um modelo WED-flow é formalmente denotado pela tupla $\langle \mathcal{G}', c_i, c_f \rangle$, onde $\mathcal{G}'$ é o conjunto de WED-triggers e c_i, c_f, são respectivamente as condições inicial e final do modelo.

3. Compilador WED-SQL

O procedimento de compilação proposto neste trabalho é composto por quatro objetos. Dois deles são objetos das classes automaticamente geradas pela ferramenta ANTLR[Parr 2014] para a análise léxica e sintática. Os outros dois objetos são responsáveis por realizar verificações de consistência das definições especificadas e por gerar o código na linguagem SQL. A interface *listener* gerada automaticamente pela ANTLR permite especificar o código de tratamento para executar após a análise sintática. Assim, a interface *listener* é usada para chamar os métodos das classes de análise semântica e de tradução para a linguagem SQL.

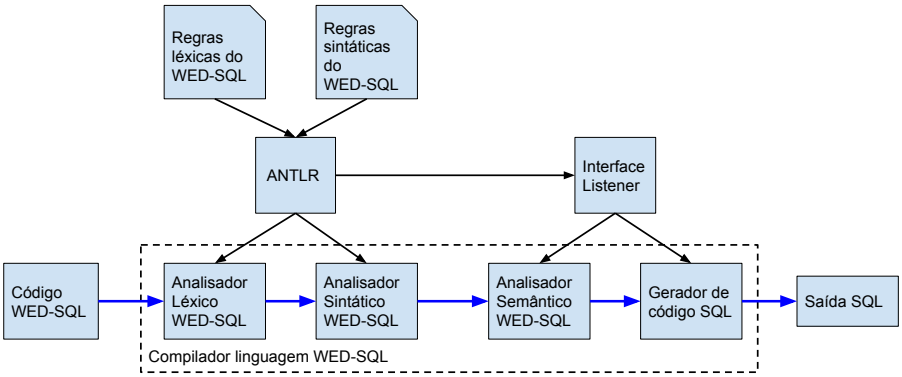

Figura 1. Arquitetura do compilador da linguagem WED-SQL.

A Figura 1 ilustra a arquitetura do compilador. Os comandos expressos na linguagem WED-SQL serão fornecidos como entrada do compilador, sendo processados pelos analisadores léxico, sintático e semântico. Os dois primeiros analisadores são automaticamente gerados a partir das regras expressas na gramática da linguagem. O analisador semântico é escrito manualmente fazendo uso da interface *Listener* gerada também pelo ANTLR a partir da descrição da linguagem. Por fim, o gerador de código na linguagem alvo, também escrito manualmente fazendo uso da interface *listener*, gera a saída do compilador em linguagem SQL.

3.1. Análise Léxica

Os analisadores léxicos gerados por ANTLR seguem o princípio da máxima correspondência (*Maximal Munch Principle*)[Parr 2013], o que significa que o analisador sempre tenta validar a maior entrada possível em uma única regra léxica. Seguem exemplos de definições das regras léxicas da gramática que geram a linguagem WED-SQL, conforme Código 1.

Código 1. Exemplos de regras léxicas que definem tokens usados pela linguagem WED-SQL.

```
CREATE : [Cc] [Rr] [Ee] [Aa] [Tt] [Ee];
WED_FLOW : [Ww] [Ee] [Dd] '–' [Ff] [Ll] [Oo] [Ww];
WED_ATTRIBUTE : [Ww] [Ee] [Dd] '–' [Aa] [Tt] [Tt] [Rr] [Ii] [Bb] [Uu] [Tt] [Ee];
WED_CONDITION : [Ww] [Ee] [Dd] '–' [Cc] [Oo] [Nn] [Dd] [Ii] [Tt] [Ii] [Oo] [Nn];
WED_TRANSITION : [Ww] [Ee] [Dd] '–' [Tt] [Rr] [Aa] [Nn] [Ss] [Ii] [Tt] [Ii] [Oo] [Nn];
WED_TRIGGER : [Ww] [Ee] [Dd] '–' [Tt] [Rr] [Ii] [Gg] [Gg] [Ee] [Rr];
```

3.2. Análise Sintática

O algoritmo de análise sintática gerado é capaz de detectar qualquer sequência de comandos WED-SQL. Um exemplo de regras sintáticas é apresentado no Código 2.

Código 2. Exemplo de regras sintáticas dos comandos da linguagem WED-SQL.

```
wsql: stmt_list;

stmt_list: SPACE* stmt SPACE* stmt_list
         | SPACE* stmt SPACE*;

stmt: create_flow_stmt SPACE* SEMICOLON
    | create_attr_stmt SPACE* SEMICOLON
    ...

create_flow_stmt: CREATE SPACE+ FLOW SPACE+ valid_name;
```

3.3. Análise Semântica

A análise semântica foi implementada como uma classe Python com métodos que realizam as seguintes verificação de consistência de modelo:

- Existência da WED-condition e WED-transition que compõem uma WED-trigger.
- Existência de WED-condition quando é informada na criação de uma WED-transition uma condição de parada.
- Existência dos WED-attributes que definem o estado inicial de uma instância.

3.4. Tradução para SQL

O Código 3 exemplifica como é implementada a tradução para SQL. Exibindo o método responsável pela tradução do comando WED-SQL de criação de WED-trigger para a linguagem SQL. O método recebe como entrada um objeto "ctx" que representa a árvore sintática construída para validar o comando de criação de WED-trigger e retorna o comando SQL resultante e os parâmetros usados no comando.

Código 3. Tradução do comando de criação de WED-trigger para SQL.

```
def create_trigger(self, ctx):
    trigger_id = ctx.valid_name().getText()
    trigger_expr = ctx.trigger_expr()
    cond_id = trigger_expr.valid_name(0).getText()
    trans_id = trigger_expr.valid_name(1).getText()
    stmt = "INSERT INTO wed_trig (tgname, cname, trname) VALUES (%s,%s,%s);"
    params = (trigger_id, cond_id, trans_id,)
    return (stmt, params)
```

4. Compilador da linguagem WED-flow

A linguagem de modelagem WED-flow oferece comandos definir os elementos de um modelo de acordo com as definições da abordagem, sem especificar detalhes de implementação. Assim, a linguagem WED-flow reduz a carga semântica da modelagem

em comparação à linguagem WED-SQL, facilitando a representação dos elementos do modelo.

A linguagem WED-SQL é uma linguagem intermediária entre a notação formal de conjuntos do WED-flow e a linguagem SQL. Assim, linguagem WED-flow é traduzida para linguagem WED-SQL. Essa tradução é viável, pois a linguagem WED-SQL contém os comandos de especificação equivalentes aos comandos oferecidos pela linguagem WED-flow.

O compilador da linguagem WED-flow segue a mesma arquitetura do compilador de linguagem WED-SQL. Faz uso da ferramenta ANTLR para gerar os analisadores léxico e sintático a partir de descrições formais da linguagem WED-flow e auxiliar a implementação das demais etapas de compilação.

5. Demonstração

Este trabalho apresenta a modelagem WED-flow do sistema SISAUT [Ferreira et al. 2017]. A Universidade de São Paulo oferece um serviço público de realização de autópsias na cidade de São Paulo, o SVOC (Serviço de Verificação de Óbitos da Capital), que permite que os órgãos do falecido sejam doados para projetos de pesquisa científica caso haja consentimento da família do falecido. O sistema SISAUT gerencia os processos envolvidos na doação de órgãos para fins de pesquisa. O SISAUT possui três processos de negócio, os quais serão modelados seguindo a abordagem WED-flow. Os modelos serão implementados no ambiente transacional e exemplos de instâncias serão gerados para a análise da execução dos processos.

O modelo de cada processo é descrito na linguagem WED-flow e fornecido ao compilador de linguagem WED-flow para gerar o código WED-SQL equivalente ao modelo. Depois disso, utilizando o compilador de linguagem WED-SQL, o código SQL é gerado para ser executado no ambiente transacional da implementação WED-flow, criando um banco de dados que contém o modelo em questão.

As WED-transitions dos processos foram implementadas na forma de serviços Python seguindo o modelo de WED-worker, apresentado em [Padilha 2018]. As WED-transitions que compõem o sistema SISAUT, além de alterarem do estado de dados de uma instância, podem iniciar diversos procedimentos complexos, manuais ou automatizados, realizados por múltiplos atores.

Um video demonstrativo das ferramentas apresentadas neste trabalho está disponível em `https://drive.google.com/file/d/1E_YSaRiK34gstcE3RR3aUcaa3RHBXGGV/view`

6. Conclusão

Este trabalho contribui para modelagem, implementação e evolução de Sistemas de Informação Cientes de Processos. Mais concretamente, disponibiliza compiladores para a linguagem WED-flow de modo a tornar o processo de especificação, instanciação e monitoramento de processos mais flexível e preciso. Os compiladores possuem duas limitações, que serão assumidas como ações futuras para a evolução do trabalho ora apresentado.

- As etapas de verificação semântica e de geração de código na linguagem alvo continuam a ser escritas manualmente, já que não podem ser automatizadas pela ferramenta ANTLR, embora tenham a implementação facilitada pela biblioteca da ANTLR.
- O algoritmo de análise sintática ALL(*) gerado pela ferramenta ANTLR restringe as gramáticas das linguagens a evitarem regras indiretamente recursivas à esquerda.

Referências

Bergstra, J., Ponse, A., and Smolka, S. (2001). *Handbook of Process Algebra*. Elsevier Science, 1st edition.

Dumas, M., van der Aalst, W. M., and ter Hofstede, A. H. (2005). *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley, 1st edition.

Ferreira, J. a. E., Takai, O. K., Malkowski, S., and Pu, C. (2010). Reducing exception handling complexity in business process modeling and implementation: The wed-flow approach. In *Proceedings of the 2010 International Conference on On the Move to Meaningful Internet Systems - Volume Part I*, OTM'10, pages 150–167, Berlin, Heidelberg. Springer-Verlag.

Ferreira, J. E., Braghetto, K. R., Takai, O. K., and Pu, C. (2012). Transactional recovery support for robust exception handling in business process services. *2012 IEEE 19th International Conference on Web Services*, pages 303–310.

Ferreira, J. E., Takecian, P. L., Kamaura, L. T., Padilha, B., and Pu, C. (2017). Dependency management with wed-flow techniques and tools: a case study. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 379–388. IEEE.

Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al. (2007). Web services business process execution language version 2.0. *OASIS standard*, 11(120):5.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Padilha, B. (2018). Wed-sql: uma linguagem declarativa intermediária com apoio transacional para a modelagem e implementação de sistemas de informação cientes de processos. Master's thesis, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil.

Padilha, B., Roberto, R. L., Schwerz, A. L., Pu, C., and Ferreira, J. E. (2018). Wed-sql: An intermediate declarative language for pais execution. *2018 International Conference on Web Services (ICWS)*, pages 407–421.

Parr, A. . T. (2014). Antlr. `http://www.antlr.org`. Último acesso em 03/07/2021.

Parr, T. (2013). *The Definitive Antlr 4 Reference*. Pragmatic Bookshelf, 2nd edition.

vom Brocke, J., Baier, M.-S., Schmiedel, T., Stelzl, K., Röglinger, M., and Wehking, C. (2021). Context-aware business process management. *Business & Information Systems Engineering*.