Evento: XXV SEMINÁRIO DE INICIAÇÃO CIENTÍFICA

ANÁLISE DO MOTOR DE EXECUÇÃO DA TECNOLOGIA GUARANÁ[1]
ANALYSIS OF THE RUNTIME ENGINE OF GUARANÁ TECHNOLOGY

Ivan E. M. Kühne[2], Rafael Z. Frantz[3]

[1] Projeto de Iniciação Científica desenvolvido no Grupo de Pesquisa em Computação Aplicada, pertencente ao Departamento de Ciências Exatas e Engenharias.
[2] Aluno do curso de Ciência da Computação da UNIJUÍ, Bolsista PIBIC/CNPQ, oitentaetres@gmail.com
[3] Professor Doutor do Departamento de Ciências Exatas e Engenharias, Orientador, rzfrantz@unijui.edu.br

INTRODUÇÃO

Conforme Frantz (2012) apud Messerschmitt e Szyperski (2005), as organizações típicas trabalham com ecossistemas de software que consistem de uma variedade de aplicações que apoiam os seus processos de negócio. Frequentemente, aparecem novos processos de negócio que precisam ser apoiados por duas ou mais aplicações e os processos já existentes precisam ser otimizados, o que requer a interação com outras aplicações. Entretanto, geralmente essas aplicações não são criadas tendo em vista a sua futura integração.

Assim, foi criada, dentro da Engenharia de Software, o campo da Integração de Soluções Empresariais (EAI), que busca oferecer metodologias, técnicas e ferramentas para fazer com que diferentes aplicações, que não foram desenvolvidas com o propósito de trabalharem juntos, possam se comunicar e/ou trabalhar conjuntamente. Segundo Frantz et al. (2011), a EAI deve manter os dados e aplicações em sintonia, além de proporcionar que novas funcionalidades sejam criadas sobre as já existentes.

Entre as ferramentas desenvolvidas nesses campo, está o uso de linguagens de domínio específico (DSL) para a modelagem conceitual das soluções de integração. Conforme Ghosh (2011), as DSLs são semelhantes às linguagens de programação tradicionais. Entretanto, cada DSL é criada para utilização em um determinado domínio de problema, estando limitada a ele. Dessa forma, ela não pode ser utilizada para a resolução de problemas em outros domínios, como ocorre com as linguagens de programação de propósito geral.

Uma das DSLs desenvolvidas no campo da EAI foi a Guaraná DSL. Essa linguagem foi criada com o objetivo de possibilitar a modelagem de soluções de integração em um alto nível de abstração, oferecendo suporte para os padrões de integração documentados por Hohpe e Woolf (2004). A partir da linguagem Guaraná DSL, foi desenvolvido um motor de execução que permite que as soluções modeladas sejam transformadas em soluções computacionais reais. Inicialmente, foi desenvolvida uma versão acadêmica como prova de conceito. Esse desenvolvimento foi realizado através da implementação em linguagem Java dos componentes da sintaxe abstrata da Guaraná DSL. O desenvolvimento do motor de execução possibilitou a criação da tecnologia Guaraná, que hoje conta com uma versão comercial.

O objetivo do presente trabalho é estudar a Interface de Programação de Aplicativos (API) da segunda geração da versão acadêmica do motor de execução da tecnologia Guaraná. Para tanto, foi selecionado um dos exemplos que acompanham a documentação do código-fonte da implementação do motor de execução, com objetivo de transformá-lo em uma solução computacional real.

METODOLOGIA

O trabalho desenvolvido foi precedido das atividades realizadas como bolsista de Iniciação Tecnológica e Inovação no período de 2015 a 2016, onde houve o primeiro contato com o campo da EAI. Nessa etapa, foi necessária uma revisão bibliográfica esse assunto, uma vez que ele não é abordado no curso de graduação em Ciência da Computação, nem mesmo dentro dos componentes curriculares relativos à Engenharia de Software.

Para tanto, foram consultados livros e artigos científicos sobre EAI. Em seguida foi revisada a bibliografia relativa às DSLs, devido à sua importância na modelagem de soluções de integração. Em especial, foi estudada a linguagem Guaraná DSL, o que possibilitou a realização do restante do trabalho. Já no período de 2016 a 2017, houve o trabalho específico relacionado à análise da API da segunda geração da versão acadêmica do motor de execução. O exemplo escolhido para subsidiar esse trabalho foi o da cafeteria, que, segundo Frantz (2012) apud Hohpe (2005), é o padrão de facto para a comparação de soluções de integração do ponto de vista prático.

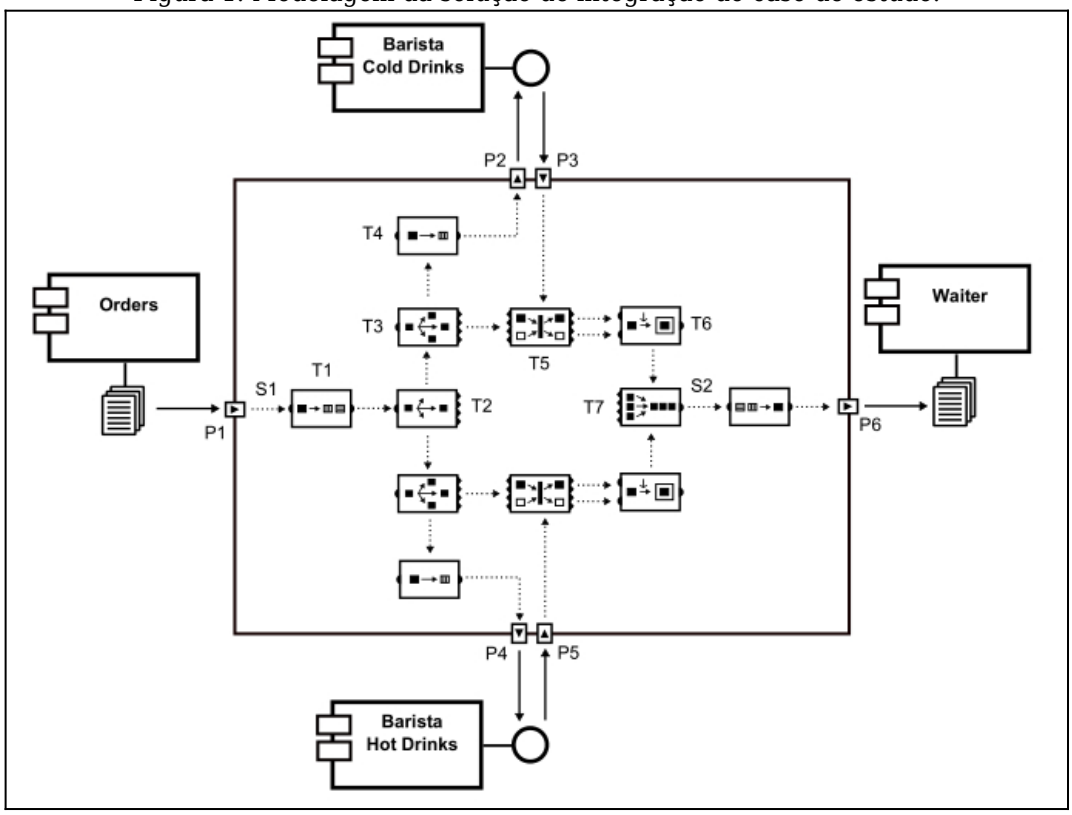
Esse problema simula uma cafeteria na qual é necessário processar os diversos pedidos que são feitos pelos clientes. Eles podem pedir bebidas quentes, como café, ou frias, como refrigerantes. Cada pedido pode, inclusive, incluir os dois tipos de bebidas. Cada tipo de bebida é peparado por um barista diferente. Quando elas estão prontas, elas são entregues ao garçom. Quando o garçom recebeu todas as bebidas relativas a um determinado pedido, ele as entrega ao cliente.

A Figura 1 representa a solução de integração proposta por Frantz (2012) para o problema da cafeteria, modelada através da linguagem Guaraná DSL. O protótipo da implementação dessa solução de integração se encontra na documentação do código-fonte do motor de execução do Guaraná. Entretanto, para que esse protótipo possa ser transformado em uma solução computacional funcional, são necessários alguns passos, como a instalação e configuração de um sistema gerenciador de um banco de dados (SGBD) compatível, que fica responsável pelo processamento externo das mensagens.

RESULTADOS E DISCUSSÃO

Foi alcançada uma implementação parcial do protótipo, sem que fosse atingida a sua funcionalidade plena. Entretanto, mesmo essa implementação parcial possibilitou o entendimento sobre alguns dos mecanismos relacionados à API do motor de execução. A classe principal é responsável pela criação de um Processo, que é o componente da linguagem Guaraná DSL que representa uma solução de integração.

Figura 1. Modelagem da solução de integração do caso de estudo.



Fonte: Frantz (2012).

Conforme o trecho de código abaixo, pode-se ver que esse processo é composto de diversos componentes, que são configurados mais adiante:

```
EntryPort ordersEntry;
SolicitorPort coldDrinksSolicitor, hotDrinksSolicitor;
ExitPort waiterExit;
Splitter splitter;
Distributor distributor;
Merger merger;
Aggregator aggregator;
ExitGate waiterExitGate, coldDrinksExitGate, hotDrinksExitGate;
IEntryPipeline ordersEntryPipeline, coldDrinkSolicitorEntryPipeline,
        hotDrinkSolicitorEntryPipeline;
IExitPipeline waiterExitPipeline, coldDrinkSolicitorExitPipeline,
        hotDrinkSolicitorExitPipeline;
InitialContext initialContext;
MysqlConnectionPoolDataSource datasource;
String ordersEntryUri, waiterExitUri, drinksSolicitorUri;
Engine engine;
```

A seguir, foram selecionados alguns trechos de código que representam a criação e configuração de alguns elementos específicos do processo. O trecho abaixo demonstra a declaração e criação de dois objetos. O primeiro deles é uma pipeline responsável por receber as mensagens do sistema. O segundo objeto corresponde à porta de entrada das mensagens no processo (representada na Figura 1 pela porta P1), que é ligada àquela pipeline no momento da sua criação.

```
IEntryPipeline ordersEntryPipeline;
ordersEntryPipeline = new OrdersEntryPipeline();
EntryPort ordersEntry;
ordersEntry = new EntryPort(
        new URI(
                "folder",
                ordersEntryUri,
                null),
        engine,
        ordersEntryPipeline);
```

O trecho a seguir demonstra a declaração e a criação de dois objetos, sendo que um corresponde a uma tarefa do tipo Splitter (representada na Figura 1 pela tarefa T1) e outro corresponde a uma tarefa do tipo Distributor (representada na Figura 1 pela tarefa T2), configurada para trabalhar com duas saídas.

```
Splitter splitter;
splitter = new Splitter(
        "/cafe_order/drinks/drink",
        "/cafe_order/order_id");
Distributor distributor;
distributor = new Distributor(2);
```

A partir da análise dos passos necessários para a configuração dos componentes da API, constatou-se que ele ocorre de maneira muito simples quando já se possui a solução de interação modelada de forma conceitual. Quando isso ocorre, basta que sejam criados os objetos referentes aos diversos componentes da solução modelada e que eles sejam configurados para trabalhar conjuntamente.

CONSIDERAÇÕES FINAIS

Como prosseguimento do trabalho, pretende-se terminar a implementação da solução computacional, de forma que a sua análise possa subsidiar o entendimento do mecanismo de planejamento e execução de tarefas, que são disponibilizados pela classe Executor. Esse entendimento deverá servir de base para o prosseguimento dos estudos no Programa de Pós-Graduação em Modelagem Matemática, após a conclusão da graduação.

Também como prosseguimento do trabalho, pode-se criar portas “mock”, que são portas que simulam a comunicação com aplicações externas. Dessa forma, pode-se usar o motor de execução

para a simulação de integração de ecossistemas de software.Essas portas podem ser criadas através do mecanismo de herança associado à Programação Orientada a Objetos, extendendo a classe Port ou SolicitorPort, ambas já existentes na API do motor de execução.

Dessa forma, pode-se usar o motor de execução para a simulação de integração de ecossistemas de software. Nessa simulação, pode-se variar o fluxo de entrada de mensagens de forma que possa ser observado o comportamento da solução de integração em determinados cenários. Assim, podem ser identificadas as características de funcionamento dessa solução antes da sua implementação real. Isso pode levar à detecção precoce de características indesejáveis, como gargalos de desempenho e tempos de processamento muito elevados, especialmente quando se trata de cenários críticos.

Palavras-chave: Integração de Aplicações Empresariais; Linguagem de Domínio Específico; Interface de Programação de Aplicativos.

Keywords: Enterprise Integration Aplication; Domain-Specific Language; Aplication Programming Interface.

AGRADECIMENTOS

Esta pesquisa foi financiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), através de bolsa de Iniciação Científica concedida pelo Processo nº 123649/2016-1.

REFERÊNCIAS

FRANTZ, R. Z.. Enterprise Application Integration: An Easy-to-Maintain Model-Driven Engineering Approach. Tese de Doutorado, Universidad de Sevilla, 2012.

FRANTZ, R. Z.; QUINTERO, A. M. R.; CORCHUELO, R.. A domain-specific language to design enterprise application integration solutions. International Journal of Cooperative Information Systems, 20(02):143–176, 2011.

GHOSH, D.. DSLs in Action. Manning Publications Co., 2011.

HOHPE, G.. Your coffee shop doesn’t use two-phase commit [asynchronous messaging architecture]. IEEE Software, 22(02): 64–66, 2005.

HOHPE, G.; WOOLF; B.. Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, 2004.

MESSERSCHMITT, D.; SZYPERSKIC, C.. Software ecosystemm: Understanding an indispensable technology and industry . MIT Press, 2003