Geração de Aplicações Situacionais a partir de Modelos de Processos de Negócio

Flávio Faria[1], Leonardo Guerreiro Azevedo[1,2], Flávia Maria Santoro[1]

[1]Programa de Pós-Graduação em Informática – Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
[2]IBM Research - Brasil

{flavio.faria, azevedo, flavia.santoro}@uniriotec.br, lga@br.ibm.com

Abstract. *Situational applications are developed by end users to solve day-to-day problems in the business scenario. The demand for this type of application is increasing since IT departments usually are not able to delivery it in agile way. Traditional software development processes do not have the agility and simplicity to fulfill the requirements of this type of application. This work presents a proposal of architecture for situational application generation using BPM, SOA and Mashup.*

Resumo. *Aplicações situacionais são aplicações desenvolvidas por usuários finais para resolver problemas do dia-a-dia no cenário coporativo. A demanda por este tipo de aplicação vem aumentando uma vez que os departamentos de TI não conseguem criar tais aplicações em tempo hábil. Os processos tradicionais de desenvolvimento de Software não são ágeis e simples o bastante para o desenvolvimento deste tipo de aplicação. Este trabalho apresenta uma proposta de arquitetura para a geração de aplicações situacionais usando BPM, SOA e Mashup.*

1. Introdução

Frequentemente, em uma organização, é necessário construir uma aplicação para atender a uma necessidade imediata do negócio (Daniel *et al*., 2012). Este tipo de aplicação é denominada aplicação situacional (Jhingran, 2006). Devido ao caráter emergencial, normalmente, é difícil desenvolvê-la utilizando os processos tradicionais para desenvolvimento de software (Pahlke *et al.*, 2010).

Normalmente, uma aplicação situacional é desenvolvida para atender a uma demanda específica de um grupo pequeno de usuários, a qual não é atendida pelos sistemas existentes na organização (Balasubramanian *et al.,* 2008). São exemplos de aplicações situacionais, aplicação para o agendamento de férias e para a gestão de seminários (Xie *et al.,* 2009). Muito embora exista uma série de trabalhos capazes de gerar uma aplicação situacional (Siemmen *et al*, 2008; Bozzon *et al*., 2009; Daniel *et al.,* 2010; Cappiello *et al.,* 2011; Truptil *et al.*, 2011; Daniel *et al*., 2012), observa-se que ainda não existe uma solução que permita o desenvolvimento de aplicações que não sejam triviais e que seja simples o bastante para ser utilizada por usuários sem conhecimento de programação (Daniel *et al.*, 2012).

Este trabalho apresenta uma arquitetura para o desenvolvimento de aplicações situacionais nas áreas de BPM (Business Process Management) e SOA (Service-

Oriented Architecture). A integração BPM e SOA têm se mostrado uma estratégia para o alinhamento entre os processos de negócio da organização e a TI (Neubauer, 2009), servindo como paradigmas de base para a construção de aplicações baseadas em serviços. Serviços são elementos computacionais auto-contidos, independentes de plataforma e que suportam composição, fundamental para o desenvolvimento de aplicações em SOA (Papazoglou *et al.,* 2007).

A principal contribuição da proposta é fornecer uma solução que combina conceitos e tecnologias existentes para produzir uma ferramenta capaz de gerar aplicações onde a lógica é derivada do fluxo de controle de um modelo de processo, tratando aspectos como transformação de modelos, mapeamento de dados, composição de serviços e execução da aplicação, em uma ferramenta orientada para o usuário final. O desafio é eliminar o conhecimento técnico, de modo que um usuário, sem conhecimento de programação, seja capaz de gerar uma aplicação.

O restante deste trabalho está organizado da seguinte forma. A seção 2 apresenta uma visão comparativa com trabalhos relacionados. A Seção 3 apresenta a arquitetura para geração de aplicações situacionais. A Seção 4 discute aspectos referentes à implementação da arquitetura. A Seção 5 apresenta a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Os principais trabalhos relacionados à proposta são apresentados a seguir.

Siemmen *et al*. (2008) propõem uma solução, chamada Damia, capaz de suportar um grande número de fontes de dados - estruturadas e não-estruturadas. A solução permite que usuários de negócio criem suas próprias aplicações de forma simples e rápida, através de uma interface gráfica onde é possível especificar e criar a aplicação automaticamente. No entanto, apesar da aplicação ser gerada automaticamente, a vinculação dos serviços que implementam as funcionalidades da aplicação é feita de forma manual. Isto exige conhecimento de padrões de desenvolvimento e programação.

Bozzon *et al*. (2009) propõem uma abordagem de desenvolvimento com base em métodos de engenharia de software como MDD (Model-Driven Development). A notação BPMN (http://www.omg.org/spec/BPMN/2.0/Beta2/PDF/) foi utilizada para especificação da aplicação. O modelo BPMN é transformado para o modelo específico de domínio, representado na linguagem WebML (Web Modeling Language) (http://www.webml.org/webml/page1.do), que é utilizado para representar a aplicação. O modelo WebML deve ser refinado antes de gerar a aplicação, inclusive, para vincular os serviços ao modelo. Apesar da geração da aplicação a partir de um modelo de processo ser abordada neste trabalho, a vinculação dos serviços à aplicação é feita de forma manual, além de ajustes relacionados à lógica de controle da aplicação. Logo, este trabalho requer do usuário conhecimentos avançados de programação.

Daniel *et al*. (2010) propõem uma solução chamada Marcoflow, para orquestração de componentes da camada de apresentação. Um conjunto de geradores de código é responsável pela geração da aplicação. Serviços são usados para fornecer o acesso aos dados e definir a lógica da aplicação. Apesar de ter capacidade de gerar a aplicação, o trabalho não abstrai detalhes técnicos ao ponto de ser considerada uma ferramenta orientada para o usuário final. A descoberta de serviço é feita manualmente, além de fazer uso de padrões complexos, como BPEL - *Business Process Execution*

Language (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel), que exige conhecimentos avançados de programação.

Cappiello *et al.* (2011) propõem uma ferramenta, chamada Dashmash, que explora tanto as fontes internas da organização quanto recursos na web, através de APIs públicas. A ferramenta oferece um ambiente gráfico para a criação e execução da aplicação, de acordo com a lógica derivada dos serviços que a compõem. A plataforma suporta ainda a descoberta automática de serviços, abstraindo os detalhes de integração, necessários para compor a aplicação. No entanto, a proposta é baseada em padrões proprietários, o que dificulta a integração com outros sistemas e a sua evolução.

Truptil *et al.* (2011) propõem uma plataforma capaz de gerar uma aplicação a partir de um modelo de processo descrito em BPMN. A linguagem BPEL é utilizada para execução do processo. A arquitetura proposta é fortemente orientada a processo, pois a lógica da aplicação é derivada do modelo BPMN. No entanto, a descoberta e vinculação dos serviços é feita de forma manual e envolve conhecimento de padrões de integração, protocolos de comunicação e configuração de elementos de infraestrutura, inviabilizando a criação da aplicação por usuários sem conhecimento de programação.

Desta forma, a proposta apresentada neste artigo tem o intuito de prover uma arquitetura para a geração automática de aplicações situacionais, por usuários sem conhecimento de programação, onde grande parte do conhecimento necessário para a geração da aplicação é derivado do modelo de processo.

3. Arquitetura para geração de aplicações situacionais

A arquitetura considera os seguintes requisitos para criação da aplicação: (i) Importar o modelo de processo e transformá-lo para um formato que possa ser analisado por um programa de computador; (ii) Extrair as características do modelo de processo, através da análise do fluxo e dos elementos (por exemplo, tarefas, eventos e gateways), que serão utilizados para definir a lógica da aplicação; (iii) Transformar o modelo de processo para um formato executável; (iv) Prover um ambiente de execução integrado para execução da aplicação gerada. Os elementos que compõem a arquitetura (Figura 1) foram divididos em módulos que por sua vez foram projetados como serviços, visando simplificar e garantir maior flexibilidade para integração entre eles.

O **Módulo de Transformação** é responsável por transformar o modelo de processo para um formato executável e gerar a aplicação. Um repositório (RDA) é utilizado para armazenar informações de configurações internas e para acesso à informações externas à arquitetura. Os serviços que compõem este módulo são: (i) Serviço de Importação do Processo (SIP): responsável por importar e extrair as características do modelo de processo; (ii) Serviço de Exportação do Processo (SEP): responsável pela descoberta dos serviços envolvidos na composição; (iii) Serviço de Geração da Aplicação (SGA): responsável por gerar a aplicação utilizando as informações extraídas do modelo de processos. O mapeamento dos dados ocorre a partir das entidades que estão associadas a cada um dos serviços incluídos na composição. Desta forma, cada entidade será instanciada em tempo de execução e dará origem às classes que compõem a aplicação. Aspectos como persistência de dados, lógica de controle e da interface gráfica da aplicação são consideradas. (iv) Serviço de implantação de aplicações (SIA): trata dos aspectos referentes à automação da

arquitetura como compilação, construção e implantação do código binário no módulo de execução.

O **Módulo de Serviços Externos** é responsável pela integração com serviços externos à arquitetura, por exemplo, a integração com o ambiente de descoberta de serviços.

O **Módulo de Execução** é responsável pela execução das aplicações geradas e pelo o gerenciamento do ambiente de execução.

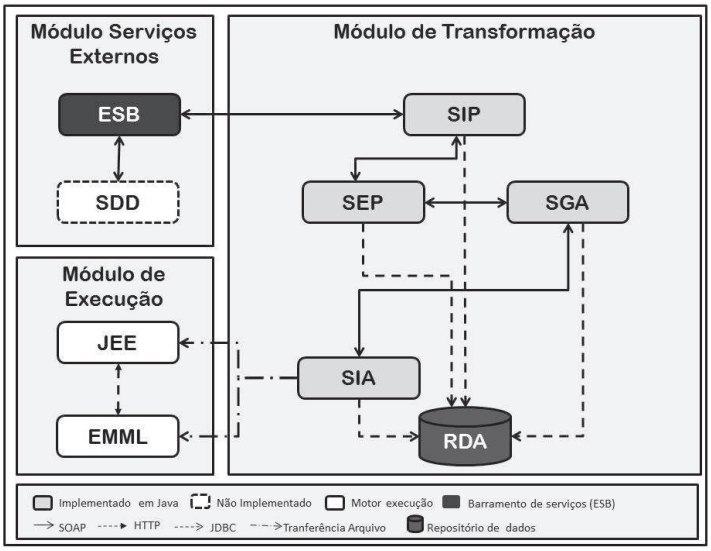

Figura 1 – Arquitetura proposta

4. Implementação da arquitetura

A arquitetura foi projetada utilizando os fundamentos de SOA. Cada módulo é composto de um ou mais serviços, com intuito de simplificar a integração entre os módulos internos à arquitetura e também com API's externas. Os serviços foram implementados utilizando a linguagem Java, API's e padrões, tais como: JAX-WS (http://jax-ws.java.net/), JAX-RS (http://jax-rs-spec.java.net/), JAXB (http://jaxb.java.net/), XML (http://www.w3.org/TR/REC-xml/), XSD (http://www.w3.org/TR/xmlschema11-1/), WSDL (http://www.w3.org/TR/wsdl20/).

BPMN foi o padrão gráfico escolhido para modelagem de processo, por ser uma notação gráfica simples para considerar os aspectos SOA (Weske, 2007). Além disso, é uma linguagem compreensível por usuários do negócio e permite o mapeamento para a linguagem de execução (Ko *et al.,* 2009).

Para representar a estrutura interna do processo BPMN foi utilizado XPDL (XML Process Definition Language), que é considerado o padrão para troca de informações entre diagramas BPMN utilizando XML (Ko *et al*., 2009). A arquitetura utiliza XPDL para importar o processo, de modo que seja possível identificar os elementos contidos no fluxo do modelo de processo.

A importação do modelo de processo ocorre a partir dos dados coletados do arquivo XPDL. Os elementos contidos no modelo de processo, bem como as transições

existentes entre os elementos, são transformados em tempo de execução em um conjunto de classes que representam o formato executável do processo.

A transformação é baseada nos padrões de workflow (van deer Aalst *et al*., 2003) que oferecem suporte adequado para a transformação (Recker e Mendling, 2006). Por exemplo, se o padrão *And Split*, que divide o fluxo do processo em dois ou mais ramos concorrentes, for identificado, então uma composição será criada para executar os serviços vinculados a cada uma das atividades envolvidas. Também é durante a transformação para a linguagem executável que ocorre a descoberta dos serviços associados às atividades do processo. O mecanismo proposto por Souza e Rabelo (2010) foi integrado à arquitetura para tratar os aspectos referentes à descoberta, onde cada atividade do processo é vinculada a um serviço através de uma ontologia baseada na especificação UBL (http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html). O mecanismo permite a integração com sistemas externos através do serviço *DiscoveryService*, que recebe como parâmetro de entrada sua classificação ontológica e retorna o *endpoint* do serviço localizado. EMML (http://www.openmashup.org/omadocs/v1.0/index.html) foi a linguagem escolhida para execução, não apenas pela simplicidade e capacidade de consumir recursos que utilizam diferentes padrões de desenvolvimento, mas também por ser considerada um padrão para o desenvolvimento de aplicações *mashup* (Cappiello *et al.,* 2011), apesar de BPEL ser considerado o padrão mais utilizado para execução do processo (Ko *et al*., 2009). A geração da aplicação ocorre em tempo de execução, com suporte nativo da linguagem Java e da biblioteca Javassist, que permite manipular *bytecode* em tempo de execução. Para cada entidade identificada durante a execução da composição, são criadas classes para representar cada camada da aplicação, por exemplo, a camada de modelo, incluindo a persistência de dados, a camada de controle e a camada de visão, que é interface do usuário. O suporte da arquitetura JAXB é necessário para permitir o mapeamento de objetos Java para XML e vice e versa, servindo como base para a geração da aplicação.

5. Conclusão

Neste artigo apresentamos uma arquitetura para o desenvolvimento de aplicações situacionais. A arquitetura proposta foi concebida utilizando padrões consolidados como SOA e BPMN e padrões emergentes como *mashup*.

O objetivo principal do nosso trabalho é fornecer uma ferramenta capaz de abstrair a complexidade para o desenvolvimento de uma aplicação SOA por usuários sem conhecimento de programação, a fim de que usuários finais desenvolvam suas próprias aplicações.

A implementação de referência da arquitetura está em andamento e já oferece suporte a padrões básicos de workflow. Alguns pontos estão sendo revistos como é o caso das condições de transição utilizadas em decisões nos padrões XOR *Split* e XOR *Join.* Com o término da implementação, o próximo passo é realizar um estudo de caso em situações reais para avaliar a real contribuição da proposta.

Referências bibliográficas

Bozzon, A., Brambilla, M., Facca, F.M., Carughui, G.F. (2009). "A Conceptual Modeling Approach to Business Service Mashup Development". In: IEEE International Conference on Web Services.

Cappiello, C., Daniel, F., Matera, M., Picozzi, M.A, Weiss, M. (2011). “Enabling End User Development through Mashups: Requirements, Abstractions and Innovation Toolkits”. In: IS-EUD , pages 9-24.

Daniel, F., Koschmider, A., Nestler, T., Roy, M., Namoun, A. (2010). “Toward Process Mashups: Key Ingredients and Open Research Challenges”. In: 3rd and 4th International Workshop on Web APIs and Services Mashups.

Daniel, F., Barkaoui, K., Dustdar, S (2012) “Business Process Management Workshops”, In: BPM 2011 International Workshops, Clermont-Ferrand, France.

Jhingran, A. (2006). “Enterprise Information Mashups: Integrating Information Simply”. In: VLDB 2006 International Conference on Very Large Databases.

Ko, R. K. L., Lee, S. S. G., Lee, E. W. (2009). “Business Process Management (BPM) Standards: A Survey”. In: Business Process Management Journal.

Neubauer, T. (2009). "An empirical study about the status of business process management". Business Process Management Journal, pages 166-183.

Pahlke, I., Beck, R. and Wolf, M. (2010). “Enterprise Mashup Systems as Platform for Situational Applications: Benefits and Challenges in the Business Domain”, In: Business and Information Systems Engineering.

Papazoglou, M. P., Traverso, P., Dutsdar, S., Leymann, F. (2007). “Service-Oriented Computing: State of the Art and Research Challenges”. In: IEEE Computer Society, vol. 40, issue 11, pages 38- 45.

Recker, J., Mendling. J. (2006). “On the Translation between BPMN and BPEL: Conceptual Mismatch Between Process Modeling Languages”. In: 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2006), June 5 - 6, 2006, Luxembourg, pages 521-532.

Truptil, S., Bartlhe, Anne-Marie., Benaben, F., Stühmer, R. (2011) “Nuclear crisis use-case management in an event-driven architecture”, In: 5th International Workshop on event-driven Business Process Management Clermont-Ferrand.

Simmen, D. E., Altinel, M., Markl, V., Padmanabhan, S., Singh, A. (2008). “Damia: Data Mashups for Intranet Applications”. In: International Conference on Management of Data (SIGMOD), pages 1171-1182.

Souza, A. P., Rabelo, R. J. (2010). " An Approach for a More Agile BPM-SOA Integration Supported by Dynamic Services Discovery". In: 14th IEEE Intl. Enterprise Distributed Object Computing Conference Workshops Pages 186-195

Van der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P. (2003) Workflow patterns. In *Distributed and Parallel Databases* vol. 14(1), p.5–51.

Xie, L., Xu, L., de Vrieze, P. (2009) “When Social Software Meets Business Process Management”, In: International Conference on Computer Sciences and Convergence Information Technology.

Weske, M. (2007) “Business Process Management: Concepts, Languages, Architectures“. Publisher: Springer. Second Edition.