Workflow Científico & WorkflowComercial

Aluno: João Antonio Ferreira, Orientador: Eduardo Ogasawara

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ

joao.parana@acm.org

Na seção seguinte apresentamos o conceito de Workflow e as iniciativas de pesquisa nesta área. Este conteúdo servirá ao propósito de *Design* de uma arquitetura para um SWfMS usando álgebra de *Workflow* como parte de uma dissertação de mestrado.

1. Fluxo de Trabalho - Workflow

Nesta seção conceituamos Worflow e itens relacionados. Fluxo de Trabalho (do inglês, Workflow) geralmente se refere ao processo de execução tanto de tarefas automatizadas, quanto manuais (com interação humana), integradas num mesmo fluxo, o qual tem por objetivo a entrega de um determinado resultado.

Nas situações em que o Fluxo de Trabalho se revela complexo se torna necessário gerencia-lo por um Sistema de Gerência de Fluxo de Trabalho (do inglês, *Workflow Management System* - WMS). Um WMS necessita de uma linguagem de especificação de Worflow. Essas linguagens podem ser: *i):* gráficas [1]; *ii):* baseadas em XML [2]; ou *iii):* baseadas em texto simples *plain-text*.

Uma sistematização e análise detalhada de diversas linguagens para especificação de Fluxo de Trabalho foi apresentada por Van Der Aalst and Ter Hofstede [2005]. Até aquele momento já existia um grande número de implementações de WMS no mercado. Os autores propuseram uma nova linguagem a qual chamaram de YAWL *Yet Another Workflow Language* com o propósito de padronizar a forma de especificar um Fluxo de Trabalho. Veja na figura 1 os símbolos usados no YAWL representando condições, tare-

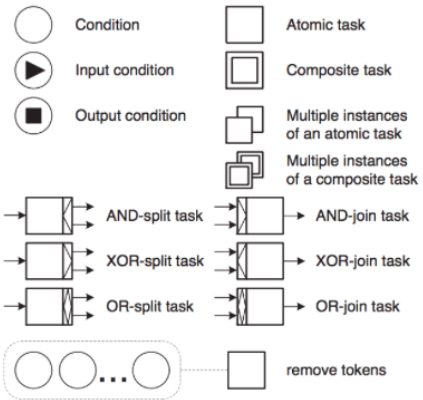

Figura 1. Elementos da linguagem YAWL

fas, e mecanismos de separação (*split*) e junção (*join*) de tarefas. A junção e separação de

[1]utilizando Interface Visual apropriada

[2]O *design* de DSL é facilitado graças a característica e**X**tensível do XML

tarefas é particularmente importante em ambiente de Cluster de Computadores onde estes estão distribuídos fisicamente e se comunicando via rede de computadores. No trabalho mencionado os autores da YAWL detalham a semântica da linguagem e demonstram a sua característica compositiva onde podemos encadear elementos de funcionalidade ortogonal (mostrados na figura 1) para obter a solução de um problema de Gerenciamento de Fluxo de Trabalho. Observe que um Workflow arbitrário que use qualquer linguagem de especificação pode ser representado por um Grafo Acíclico Direcionado - DAG onde cada vértice pode ser uma condição, uma tarefa, um mecanismo de separação ou um mecanismo de junção. Já as arestas são usadas para representar as dependências entre estes vértices. O grafo mencionado foi limitado ao tipo acíclico pela facilidade de desenvolvimento e por ter suporte nativo no Spark. Além disso, usando linguagem funcional (tal como Scala) podemos tratar uma tarefa que envolva Grafos Cíclicos como outra que use objeto Função (do inglês, *Function*) que possa encapsular o grafo cíclico numa tarefa do grafo acíclico.

Os autores da YAWL definem os requisitos para linguagens de fluxo de trabalho através de Padrões de Fluxo de Trabalho (do inglês, *Workflow Patterns*). Segundo Buschmann et al. [2007], um padrão para arquitetura de software descreve um problema de projeto recorrente particular que surge em contextos de design específicos e apresenta um esquema genérico bem comprovado para sua solução. O esquema de solução é especificado descrevendo seus componentes constituintes, suas responsabilidades e relacionamentos, e as maneiras pelas quais eles colaboram.

O conceito de Padrão de Projeto foi usado pela primeira vez por Gamma et al. [1993]. A chamada GoF (Gang of Four) foi quem primeiro catalogou sistematicamente cerca de 23 padrões de design que descrevem as menores interações recorrentes em sistemas orientados a objetos. Os padrões de projeto, como tal, proporcionam independência da tecnologia de implementação e, ao mesmo tempo, independência dos requisitos essenciais de domínio ao qual eles tentam atender.

Vê-se então que foi acertada a decisão de van Der Aalst et al. [2003], os autores da YAWL, que já vinham pesquisando e catalogando os Padrões de Fluxo de Trabalho desde 2003. Estes padrões nos servirão de base de pesquisa no que diz respeito à especificação funcional da Máquina Execução de Fluxo de Trabalho (do inglês, *Workflow Execution Engine*) na dissertação de Mestrado já citada.

Os 20 padrões mais relevantes descritos em van Der Aalst et al. [2003] foram classificados inicialmente em em seis categoria:

1. Padrões básicos de fluxo de controle. Essas são as construções básicas que permitem modelar roteamento seqüencial, paralelo e condicional.
2. Ramificação avançada e padrões de sincronização. Esses padrões permitem divisão e união de comportamento de forma dinâmica.
3. Padrões estruturais que permitem uma estrutura menos rígida.
4. Padrões envolvendo várias instâncias. Algumas vezes partes do processo têm de ser instanciadas várias vezes à partir de um template (modelo).
5. Padrões baseados no Estado da tarefa. Os sistemas típicos de fluxo de trabalho se concentram apenas em atividades e eventos e não em estados.
6. Padrões de cancelamento. A ocorrência de um evento de cancelamento pode levar ao cancelamento de outras actividades van Der Aalst et al. [2003].

Uma discussão mais detalhada desses padrões pode ser vista em van Der Aalst et al. [2003] e no site da *Workflow Patterns Initiative* - (veja na seção Referências: Wf-Patterns [2016]) criado em 2010 por Wil van der Aalst e outros colaboradores. A YAWL foi criada com vinte padrões. Uma década depois este número saltou para mais de quarenta, revelando que a comunidade que se dedica a estes estudos é bastante ativa. A implementação de código aberto de um *Workflow Management System* que usa a YAWL baseada nestes Padrões pode ser acessado no site do projeto na WEB [3]

Devemos ter em mente que o problema de Gerenciamento de *Workflow* deve ser tratado em diversas perspectivas como descrito por Van Der Aalst and Ter Hofstede [2005], que são: *i):* dados: passagem de informação e do escopo de variáveis; *ii):* recursos: alocação de tarefas e delegação; *iii):* tratamento de exceção: recuperação de erros de processamento; *iv):* fluxo de controle: separação de tarefas paralelizáveis, sincronização de tarefas executadas paralelamente com junção de resultados, escolha de caminhos dependendo do contexto, registro de transformações sobre os dados para auditoria, etc. No caso de Workflow Científico analisado por Ogasawara et al. [2011] e Marinho et al. [2012] essa auditoria pode ser entendida como Proveniência de Dados [4].

No escopo deste trabalho trataremos apenas de fluxo de controle. Esta categoria é a base de todo o resto. Como pode ser visto é a perspectiva mais extensa além de ser essencial ao funcionamento de qualquer *Workflow*.

2. A Linguagem WSL

Dado que o problema do Gerenciamento de *Workflow* foi contextualizado podemos pensar numa linguagem de domínio específico, em estilo declarativa, com trechos em linguagem natural, possa ser usada para descrever um dado Experimento que use um ou mais *Workflows*. Podemos usar palavras reservadas tais como *Experiment, Workflow, Task, Split, Join, Given, When, And, But, Mining, Method, Dataset, Input, Output, YARN* que serviriam para criar uma gramática dependente de contexto numa linguagem que podemos chamar simplesmente *Workflow Simple Language* - WSL. O propósito dessa linguagem seria a de permitir especificação de requisitos de forma que a comunicação entre os cientistas e seus clientes possa se dar objetiva e sem ambiguidades. A implementação em Scala de um *Workflow* especificado em WSL poderá ser dinâmica o suficiente para permitir a adição de métodos de mineração usando as características de Carga Dinâmica de Classes (do inglês, Dynamic Class Loading) da JVM [5]. Além disso, teríamos o benefício da própria especificação do *Workflow* forneçer uma documentação não ambígua do ambiente e das tarefas envolvidas. Um exemplo desse tipo de especificação aparece na figura 2

A WSL pode definir uma gramática simples e expressiva que pode ser usada para guiar a programação em Linguagem Scala de um coordenador de tarefas, podendo ser executada em ambiente *Apache Spark*. Para testar os programas Scala no ambiente *Spark* [6] provisionado de diversas formas, como por exemplo, num cluster de computadores geren-

[3] Veja na seção Referências: YAML [2016]

[4] O termo auditoria é aplicada no caso de Workflow de Negócios. Proveniência de Dados é aplicada no caso de Workflow Científico

[5] Dynamic class loading in the Java Virtual Machine - Liang and Bracha [1998]

[6] Veja seção Referências : Spark [2016]

```
Experiment: Finding routes
 Workflow: Mining Clustering id: Step 1 env: YARN
   Given the dataset 3D_spatial_network.csv
   Then I should get Mining results using K_Means Method as a Task
   Input for task is numMaxIterations: 20
   The Output must be Hive table with name: 3DS-kmeans
   Input condition: First sampling
     Only 5 MB and 9MB tuples acquired randomly from dataset
     | size | method | kValue |
     | 5MB  | random |     20 |
     | 9MB  | random |     10 |
```

Figura 2. Exemplo de Workflow especificado via WSL

ciados pelo YARN, como contêineres Docker ou mesmo *stand-alone* num Laptop usando o *spark-shell*.

3. *Workflow Management Coalition*

Como o problema de gerenciar fluxo de trabalho é bastante comum surgiu a necessidade de criar uma associação que cuidasse da padronização, distribuição de conhecimento, e divulgação de soluções existentes neste setor da economia. Assim foi criado a organização não governamental *Workflow Management Coalition* WfMC - (WfMC [2016]) com tal finalidade. Segundo a WfMC, Fluxo de Trabalho é a sistematização do processo de negócio, na sua totalidade ou em partes, onde documentos, informações ou tarefas são passadas de um participante para outro para execução de uma ação, de acordo com um conjunto de regras de procedimentos. Nesta definição o participante pode se referir a um indivíduo ou a um processo automatizado em um sistema. Vale lembrar também que ”processo de negócio”pode se referir a um experimento científico *in-silico*.

A WfMC tem se esforçado para padronizar uma linguagem de especificação de fluxo de trabalho, porém os principais fornecedores de WMS utilizam cada um sua própria linguagem além de conceitos baseados em diferentes paradigmas. A falta de consenso se deve não só a interesses comerciais diversos, como também a ausência de uma teoria organizacional universal no que diz respeito a padronização de conceitos de modelagem de processos de negócio Van Der Aalst and Ter Hofstede [2005].

4. *Conclusão*

Com o resumo da descrição deste método sistemático baseado em Padrões de Workflow e das vantagens de uma DSL tal como WSL encerramos este Documento.

Referências

Buschmann, F., Henney, K., and Schmidt, D. C. (2007). *Pattern-oriented software architecture, on patterns and pattern languages*, volume 5. John wiley & sons.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. In *European Conference on Object-Oriented Programming*, pages 406–431. Springer.

Liang, S. and Bracha, G. (1998). Dynamic class loading in the java virtual machine. *Acm sigplan notices*, 33(10):36–44.

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S. M. S. d., Ogasawara, E., and Mattoso, M. (2012). Provmanager: a provenance management system for scientific workflows. *Concurrency and Computation: Practice and Experience*, 24(13):1513–1530.

Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. (2011). An algebraic approach for data-centric scientific workflows. *Proc. of VLDB Endowment*, 4(12):1328–1339.

Spark (2016). Projeto apache spark. http://spark.apache.org/docs/2.0.2/. Accessed: 2016-12-09.

Van Der Aalst, W. M. and Ter Hofstede, A. H. (2005). Yawl: yet another workflow language. *Information systems*, 30(4):245–275.

van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., and Barros, A. P. (2003). Workflow patterns. *Distributed and parallel databases*, 14(1):5–51.

WfMC (2016). Worflow management coalition. http://www.wfmc.org. Accessed: 2016-12-09.

WfPatterns (2016). Workflow patterns home page. http://www.workflowpatterns.com. Accessed: 2016-12-09.

YAML (2016). Yawl no github. https://github.com/yawlfoundation/yawl. Accessed: 2016-12-09.