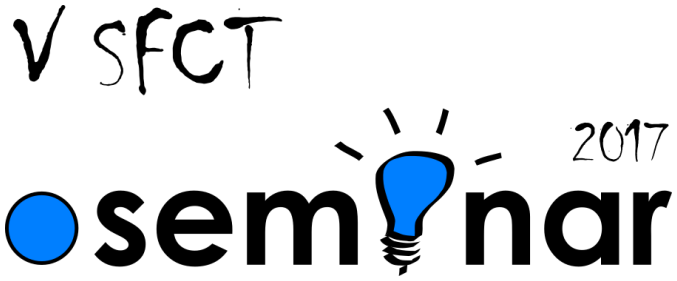

V Seminário de Formação Científica e Tecnológica

Grupo de Pesquisa em Computação Aplicada

www.gca.unijui.edu.br

24 – Abril – 2017
9:00 – 12:00 e 13:30 – 17:00

Local:
Departamento de Ciências Exatas e Engenharias - DCEENG
Ijuí, RS. Brasil

Índice

V SFCT

O V Seminário de Formação Científica e Tecnológica é um evento promovido pelo Grupo de Pesquisa em Computação Aplicada (GCA) e tem como objetivo criar um espaço local de debate que possa contribuir positivamente sobre o trabalho que vem sendo desenvolvido pelos membros do grupo, especialmente alunos de graduação e pós-graduação.

Coordenação Geral:

Dr. Rafael Z. Frantz
UNIJUÍ, Brasil

Comitê Científico:

Dra. Fabricia Roos-Frantz
UNIJUÍ, Brasil

Dra. Inma Hernandez
Universidad de Sevilla, Espanha

Dra. Iryna Yevseyeva
Newcastle University, Reino Unido

Dr. Michael Emmerich
University of Leiden, Holanda

Dr. Rafael Corchuelo
Universidad de Sevilla, Espanha

Dr. Rafael Zancan Frantz
UNIJUÍ, Brasil

Dr. Carlos R. Osuna
Rochester Institute of Technology, United States

Dr. Sandro Sawicki
UNIJUÍ, Brasil

Dr. Vítor Manuel Basto Fernandes
Instituto Politécnico de Leiria, Portugal

Comitê de Organização:

Daniela Freire Sellaro
UNIJUÍ, Brasil

Eldair Fabrício Dornelles
UNIJUÍ, Brasil

Igor G. Haugg
UNIJUÍ, Brasil

Matheus Henrique Rehbein
UNIJUÍ, Brasil

Uma Proposta Metodológica para a Simulação de Soluções de Integração de Aplicações Empresariais

Arléte Kelm Wiesner
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brasil
arlete.kelm@gmail.com

***Resumo*—A área de integração de aplicações empresariais proporciona metodologias e ferramentas para projetar e implementar soluções de integração. Algumas das ferramentas são Camel, Spring Integration, Mule e Guaraná. A análise de soluções de integração, para prever seu comportamento e encontrar possíveis gargalos de desempenho, é uma importante atividade que contribui para aumentar a qualidade das soluções desenvolvidas. A abordagem, geralmente adotada pelos engenheiros de software, consiste na construção e execução da solução de integração. No entanto, o desenvolvimento da solução envolve custos e riscos inerentes, que geralmente são elevados. Soluções de integração podem ser classificada como sistemas estocásticos, dinâmicos e discretos, podendo assim ser simulados, por meio de técnicas e ferramentas para simulação de eventos discretos. Nesse contexto, este artigo aborda uma proposta de metodologia para auxiliar no processo de simulação de soluções de integração empresariais.**

Palavras-chave: Integração de Aplicações Empresariais; Simulação; Metodologia.

I. Introdução

As empresas adquirem ou desenvolvem aplicações para apoiar a tomada de decisões, a coordenação, o controle e o aperfeiçoamento de seus processos de negócio. Essas aplicações compõe o ecossistema de software da empresa, que geralmente é heterogêneo. Normalmente, as aplicações que compõem o ecossistema de software, foram projetadas sem levar em conta a possibilidade de serem integradas. A integração de aplicações é importante, porque permite a reutilização de duas ou mais aplicações para apoiar novos processos de negócio, ou para otimizar processos já existentes.

Atualmente, existem várias tecnologias de integração que oferecem suporte a concepção e implementação de soluções de integração. Camel [1], Spring Integration [2], Mule [3] e Guaraná [4], são algumas das tecnologias que proporcionam uma Linguagem de Domínio Específic (DSL) baseada nos padrões de integração [5] com suporte para projeto, desenvolvimento e execução de soluções de integração. Essas linguagens seguem o estilo arquitetônico de *Pipes and Filters.* Nesse estilo, um processo é dividido em uma sequência de processos menores e independentes (*Filters*), que são conectados por canais (*Pipes*).

O sucesso das empresas em seus processos de negócios, atualmente, depende da execução correta e eficient das soluções de integração. Portanto, a análise de soluções de integração de aplicações empresariais, para prever seu comportamento com diferentes cargas de trabalho e identifica possíveis gargalos de desempenho é considerada uma importante atividade para melhorar a qualidade das soluções entregues. Descobrir se uma solução de integração pode falhar e em que condições pode acontecer é uma tarefa onerosa, arriscada e demorada [6]. Isso porque normalmente, a abordagem adotada pelos engenheiros de software para análise do comportamento frente a cenários críticos de funcionamento e o recolhimento de dados demandam a construção e a execução das soluções de integração. Uma abordagem que permite analisar o comportamento e identifica possíveis gargalos de desempenho, ainda na fase de projeto, a partir do modelo conceitual, poderá reduzir custos, riscos e tempo de desenvolvimento.

Uma solução de integração pode ser caracterizada como um sistema, cujo modelo conceitual é classificad como estocástico, dinâmico e discreto [6]. Modelos discretos são orientados a eventos e usados para modelar sistemas que mudam seu estado em momentos específico no tempo, a partir da ocorrência de eventos. Soluções de integração podem ser caraterizadas como sistemas discretos, porque quando ocorre um evento todos os componentes envolvidos na solução consomem um tempo específic de execução. Como um sistema discreto, o modelo conceitual de uma solução de integração, pode ser simulado utilizando técnicas e ferramentas para a simulação de sistemas de eventos discretos.

Para que a análise de um sistema, por meio da modelagem e simulação, seja bem sucedida, é necessário seguir algumas etapas. Nesse sentido, o objetivo deste trabalho é apresentar a proposta de uma metodologia para a simulação de soluções de integração empresariais que proporcionem uma DSL, baseada nos padrões de integração seguindo a arquitetura *Pipes and Filters*.

De acordo com Paul and Balmer [7] (1993), o desenvolvimento de um modelo de simulação é composto de três grandes etapas: formulação do modelo, implementação e análise dos resultados. Para o autor, na primeira etapa deve-se entender e compreender o sistema que será simulado e desenvolver o modelo conceitual. A segunda etapa é definid como sendo a conversão do modelo conceitual em um modelo computacional, que a partir da geração de alguns resultados pela execução do modelo de simulação deve ser verificad e validado. Na terceira etapa, o modelo computacional é executado várias vezes para a geração de dados que são interpretados, analisados estatisticamente e documentados.

Van Der Aalst [8] (2015) afirm que o processo de simu-

lação começa com a definiçã do problema. Para tanto, deve-se descrever os objetivos e estabelecer o escopo do estudo da simulação. Na fase de modelagem o modelo conceitual é criado. Após a criação do modelo conceitual, inicia-se a fase de realização, na qual o modelo conceitual é transformado em um modelo executável. A forma de criação deste modelo, depende da ferramenta de simulação utilizada. Um modelo executável não está necessariamente correto, por isso, ele precisa ser verificad e validado.A partir do modelo validado, as experiências podem ser realizadas. Nesta fase, é decidido quantas execuções e a duração de cada simulação. Os resultados da simulação precisam ser interpretados para responder as perguntas da definiçã do problema.

Outros autores também abordam os métodos que devem ser utilizados para solucionar um problema por meio do processo de modelagem e simulação de sistemas. Banks et al. [9] (2005) afirm que as etapas de um estudo de simulação são as seguintes: formulação do problema, definiçã dos objetivos e plano geral do projeto, modelo conceitual, coleta de dados, tradução dos modelos, verificação validação, design experimental, execução e análise, documentação e relatórios, finalizand com a implementação. Ainda, segundo o autor, discussões semelhantes podem ser encontradas em Shannon [10] (1975) e Law et al. [11] (2000).

A partir da análise da literatura técnica fico evidente que a abordam da metodologia para modelagem e simulação de sistemas discretos tem sido amplamente discutida, contudo ainda não foi explorada no contexto da simulação de soluções de integração empresariais, tendo como entrada seus modelos conceituais.

O restante do artigo está organizado da seguinte forma: A Seção II apresenta a descrição da metodologia e a Seção III as conclusões.

II. Metodologia

O processo de modelagem e simulação de sistemas, geralmente segue um conjunto de etapas. Nesse sentido, essa Seção apresenta os resultados iniciais do desenvolvimento de uma metodologia para auxiliar no estudo e análise de soluções de integração, por meio da simulação de seus modelos conceituais. A Figura 1, mostra um conjunto de etapas para orientar o desenvolvimento do modelo de simulação.

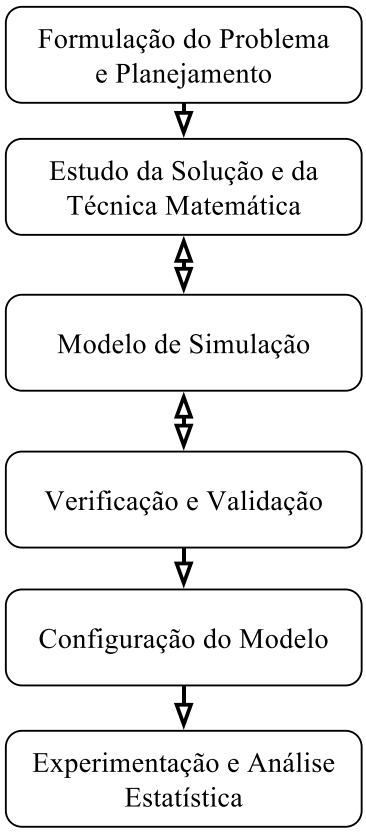

Figura 1. Passos da metodologia para simulação de soluções EAI

A. Formulação do Problema e Planejamento

O estudo de uma solução de integração, por meio da simulação de seu modelo conceitual, deve iniciar com a formulação do problema e com o planejamento do estudo. Obter as respostas certas demanda conhecimento do problema. Não é possível estudar e resolver um problema sem antes conhecê-lo profundamente. Por isso, deve-se defini de forma clara e precisa os propósitos e os objetivos da simulação.

B. Estudo da Solução e da Técnica Matemática

O conhecimento é a base para o desenvolvimento de um modelo de simulação. Desenvolver o modelo requer conhecer a solução de integração, seu flux de trabalho e funcionalidades, bem como a técnica matemática que será utilizada para avaliar as medidas de desempenho. Assim, deve existir uma interação constante entre a pesquisa e o desenvolvimento do modelo de simulação.

Os objetivos da simulação determinam, de forma ampla, as informações que serão necessárias. Nessa etapa, deve-se determinar os elementos da solução, suas interconexões ou relações, defini as entradas e saídas e demonstrar a equivalência dos elementos da solução de integração com a técnica matemática.

C. Modelo de Simulação

Nessa etapa, utilizando como base uma técnica matemática, o modelo conceitual da solução de integração é transformado em um modelo de simulação equivalente. Dessa forma, a criação do modelo dependente diretamente da técnica matemática que será utilizada, consequentemente da ferramenta de simulação.

Para desenvolver o modelo de simulação, é preciso ter a especificaçã da equivalência dos elementos da solução de integração, variáveis, parâmetros, regras e relações com a técnica matemática e ferramenta de simulação. Contudo, as ferramentas de simulação também requerem uma parametrização correta, por isso antes de iniciar o desenvolvimento do modelo de simulação, é preciso conhecer sua estrutura e funcionalidades. Os elementos do modelo conceitual da

solução de integração devem ser transcritos, considerando os blocos de construção da ferramenta e as funcionalidades que apresentam.

O processo de modelagem e simulação de uma solução de integração consiste em abstrair os aspectos essenciais do problema e selecionar os pressupostos básicos que caracterizam a solução. Assim, para desenvolver um modelo de simulação equivalente ao modelo conceitual, o ideal é começar com um modelo simples, abrangendo apenas a essência da solução de integração e posteriormente construir para uma maior complexidade.

D. Verificaçã e Validação

Durante o desenvolvimento do modelo de simulação, é necessário estar seguro de que o mesmo está sendo implementado corretamente. Isso signific estar livre de erros de sintaxe e/ou lógica e ter acurácia suficient para ser utilizado como substituto da solução de integração, para a análise e experimentação. Essas duas etapas são conhecidas como verificaçã e validação do modelo de simulação.

Para fin de verificaçã e validação, pequenos testes podem ser simulados e os resultados analisados em concordância com alguma das técnicas formais descritas na literatura. Algumas dessas técnicas foram descritas e utilizadas por Wiesner [12]. A verificaçã e validação podem levar a ajustes no modelo de simulação.

E. Configu ação do Modelo

Após ser verificad e validado, o modelo é ajustado para realizar as experimentações. Nessa etapa, são configurado os parâmetros de entrada do modelo considerando os cenários que serão estudados, o tempo de duração da simulação e o número de repetições.

Para alcançar precisão estatística sobre os resultados da simulação, é necessário repetir a execução do modelo várias vezes. Em estatística, quando um experimento é repetido um grande número de vezes com os mesmos dados, seguindo a Lei dos Grande Números [13], conforme o número de repetições se incrementa, a média amostral das variáveis do experimento se aproxima, cada vez mais, da média populacional, também conhecida como média teórica ou esperança matemática. Empiricamente, para a análise de sistemas que ainda não existem, onde para obter os dados é executado um experimento artificial a média populacional costuma ser obtida com, aproximadamente, 25 replicações.

F. Experimentação e Análise Estatística

Nessa etapa, as simulações são executadas para a geração dos dados e suas análises subsequentes usadas para estimar as medidas de desempenho da solução de integração. No entanto, toda e qualquer experimentação está sujeita a apresentar dados extremos. Na estatística, os dados que apresentam um grande afastamento dos demais (valores muito altos ou muito baixos) são conhecidos como outliers. Geralmente, os dados que representam outliers são retirados da amostra, para obter dados mais homogêneos.

Para encontrar os outliers, os resultados da simulação são submetidos ao método de Tukey [14]. Os valores que estão abaixo do limite inferior (LI) ou acima do limite superior (LS) estabelecidos são considerados outliers e removidos. Posteriormente, são calculadas as médias dos dados das 25 repetições, e construídos os gráfico das variáveis para analisar e avaliar as medidas de desempenho da solução de integração.

III. Conclusão

Para que a abordagem de uma solução de integração, baseada na simulação de eventos discretos seja bem sucedida, é necessário seguir algumas etapas. Nesse contexto, esse artigo propôs uma proposta para o desenvolvimento de uma metodologia para auxiliar e orientar no processo de simulação de soluções de integração.

Esse trabalho descreve os resultados iniciais da proposta de desenvolvimento das etapas necessárias para a simulação. Como trabalho futuro, pretende-se aprofundar o estudo do desenvolvimento da metodologia, com o intuito de especifica de maneira mais detalhada cada etapa e apresentar um caso de estudo com o objetivo de demonstrar a aplicação da metodologia. Assim, a metodologia poderá ser utilizada como um guia no desenvolvimento do estudo de simulação para encontrar gargalos de desempenho em soluções de integração a partir de seus modelos conceituais.

Referências

[1] C. Ibsen and J. Anstey, *Camel in action*. Manning Publications Co., 2010.

[2] M. Fisher, J. Partner, M. Bogoevici, and I. Fuld, *Spring Integration in action*. Manning Publications Co., 2012.

[3] D. Dossot, J. D'Emic, and V. Romero, *Mule in action*. Manning, 2014.

[4] R. Z. Frantz and R. Corchuelo, "A software development kit to implement integration solutions," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 1647–1652.

[5] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[6] S. Sawicki, R. Z. Frantz, V. M. B. Fernandes, F. Roos-Frantz, I. Yevseyeva, and R. Corchuelo, "Characterising enterprise application integration solutions as discreteevent system," in *Handbook of Research on Computational Simulation and Modeling in Engineering*. IGI Global, 2015, pp. 255–282.

[7] R. J. Paul and D. W. Balmer, *Simulation modelling*. Chartwell-Bratt, 1993.

[8] W. M. Van Der Aalst, "Business process simulation survival guide," in *Handbook on Business Process Management 1*. Springer, 2015, pp. 337–370.

[9] J. Banks, J. S. CARSON II, L. Barry *et al.*, *Discrete-event system simulationfourth edition*. Pearson, 2005.

[10] R. E. Shannon, "Systems simulation," 1975.

[11] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation modeling and analysis*. McGraw-Hill New York, 2000, vol. 2.

[12] A. K. Wiesner, "Modelagem e simulação de uma solução de integração para identificaçã de gargalos de desempenho baseadas em formalismo matemático: uma abordagem orientada à teoria das filas" 2016.

[13] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 2012.

[14] J. W. Tukey, "Exploratory data analysis," 1977.

Particle Swarm Optimization para agendamento de tarefas na integração de aplicações empresariais

Daniela F. Sellaro*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brasil
dsellaro@unijui.edu.br

***Resumo*—As empresas buscam alternativas tecnológicas que proporcionem competitividade para seus processos de negócios. Dentre essas alternativas, existem as plataformas de integração que conectam as aplicações da empresa, por meio de soluções de integração, fazendo com que elas funcionem de forma sincronizada, oferendo acesso às informações e funcionalidades de forma rápida e segura. Essas plataformas adotam algoritmos de agendamento de tarefas baseados em heurísticas, nas quais pode ocorrer uma espera inadequada das tarefas na fila para ser executadas. A eficiência no agendamento das tarefas impacta no desempenho da solução de integração e este é um dos desafios enfrentados pelas plataformas de integração. Esse artigo apresenta um algoritmo baseado na técnica de otimização *Particle Swarm Optimization*, que atribui as tarefas para os recursos computacionais, considerando o tempo de espera na fila e na complexidade computacional das tarefas, buscando minimizar o tempo total de execução da solução de integração.**

Palavras-chave: Integração de Aplicações Empresariais; Otimização; Motor de execução; Algoritmo de agendamento de tarefas.

I. Introdução

As empresas possuem um ecossistema de software composto por diversas aplicações, que geralmente são desenvolvidas internamente ou adquiridas de terceiros, o qual suporta seus processos de negócio. O avanço das tecnologias de desenvolvimento de aplicações e a incorporação de serviços de software disponíveis na internet têm deixado os ecossistemas de software heterogêneos, e geralmente, essas aplicações e serviços não estão preparados para trabalhar de forma conjunta. Porém, para oferecer respostas rápidas e consistentes aos processos de negócio, é necessário que as aplicações trabalhem de forma sincronizada. Para alcançar esse objetivo, as empresas contam com plataformas de integração, que são ferramentas de software especializadas que permitem projetar, executar e monitorar soluções de integração. Uma solução de integração é um programa computacional que conecta funcionalidades e dados de diferentes aplicações por meio de tarefas, que são artefatos executáveis, que funcionam como adaptadores [6]. As plataformas geralmente fornecem uma linguagem de domínio específico, um kit de ferramentas de desenvolvimento, um motor de execução e uma ferramenta de monitoramento. Desses, é o motor que proporciona todo o suporte necessário à execução das soluções de integração [5]. As tarefas da solução de integração são executados por pools de threads, as unidades básica de processamento da CPU, os quais compõem o motor de execução. Essas tarefas geralmente aguardam em uma fila de tarefas prontas para serem executadas, obedecendo uma ordem estabelecida por um algoritmo de agendamento de tarefas. As heurísticas mais utilizadas por esses algoritmos são a de Prioridade e a *First-In-First-Out* (FIFO). Com uma politica baseada em prioridades, níveis são atribuídos às tarefas e aquelas com um nível maior de prioridade têm mais chance de serem executadas do que aquelas de nível inferior. Já na política FIFO, as tarefas são executadas na ordem de chegada, porém quando a taxa de entrada de mensagens aumenta, costuma haver um acúmulo das tarefas iniciais na fila e as tarefas subsequentes passam a ter menos chance de serem executadas. Em ambos os casos, poderá haver um aumento no tempo de espera das tarefas na fila. Um algoritmo de agendamento de tarefas inadequado pode impactar no tempo de execução das tarefas [7], e assim degradar o desempenho da execução de uma solução de integração. Esse trabalho propõe um algoritmo, que faz a agendamento das tarefas da solução de integração para os pool de threads do motor de execução, considerando o tempo de espera na fila de tarefas e a complexidade computacional da tarefa. Ele é modelado como um problema de otimização, no qual utilizou-se a técnica metaheurística *Particle Swarm Optimization* (PSO), a qual tem sido adotada com sucesso em diferentes domínios tais como sistemas elétricos [2], inteligência computacional [10], dentre outros. PSO é usada para escalonar e agendar a execução das tarefas para os recursos computacionais, buscando minimizar o tempo de execução total do processamento da mensagem, sem aumentar o consumo dos recursos computacionais [9]. Além disso, o algoritmo define a configuração mais adequada dos recursos computacionais e o agendamento mais apropriado para a execução das tarefas, conforme a solução de integração. O resto deste artigo está organizado como segue. A Seção 2 apresenta a formulação do problema. A Seção 3 expõe a abordagem proposta. Por fim, a Seção 4 apresenta as conclusões e trabalhos futuros.

II. Formulação do Problema

O motor de execução de uma plataforma de integração é composto pools de threads, *Pool*, com diferentes números de threads, que define um tipo $Pool_i$, uma capacidade limitada

* Bolsista PROSUP/CAPES

de processamento P_{Pool_i}, definida em termos de instruções por ciclo (IPC) [1]. A variação do desempenho pode ser modelada pelo ajuste da capacidade do *Pool* e introduzindo uma degradação de desempenho $degPool_i$ [9]. O tempo de execução da tarefa em um pool é estimado por meio do tamanho da tarefa, conforme Equação 1.

$$TE_{t_i}^{Pool_j} = Tam_{t_i}/(P_{Pool_j} * (1 - \deg_{Pool_j})) \quad (1)$$

onde: Tam_{t_i} = tamanho da tarefa

P_{Pool_j} = capacidade de processamento do pool de threads

$\deg_{Pool_j}$ = degradação de desempenho.

Já o tempo total de processamento $TP_{t_i}^{Pool_j}$ de uma tarefa em um pool, considera o tempo de espera na fila de tarefas, é calculado pela Equação 2:

$$TP_{t_i}^{Pool_j} = TE_{t_i}^{Pool_j} + (\sum_{1}^{k} TFe_{ij} + s_k) \quad (2)$$

onde: $TE_{t_i}^{Pool_j}$ = tempo de execução da tarefa em um determinado pool de threads

TFe_{ij} = tempo de espera na fila de tarefas, ou seja, tempo para transferir dados entre uma tarefa pai e sua tarefa filha s_k = tempo gasto na troca de pool de threads k = número de arestas em que uma tarefa é uma tarefa pai O agendamento de tarefas é definido por $A = (R; M; TP; TTE)$, sendo R um conjunto de recurso, onde cada recurso r_i tem associado a ele: um *Pool* do tipo $Pool_i$, um tempo de início estimado para alocação do recurso estimado $TIni_{r_i}$. e um tempo de finalização estimado $TFim_{r_i}$.; M o mapeamento de tarefas em recursos, TR o total de pools e TTE o tempo total de execução, que é calculado pela Equação 3:

$$TTE = \max\{TFim_{t_i} : t_i \in T\} \quad (3)$$

Assim, o problema pode ser formulado como: *encontrar um agendamento A com o menor tempo total de execução TTE da solução de integração, sem exceder um valor pré-definido para o total de recursos TR*. Essa formulação é representada por 4:

$$Minimize\ TTE \quad (4)$$

$$\text{sujeito a } TR \leq \delta_r$$

III. Algoritmo de Agendamento de Tarefas modelado no PSO

Particle Swarm Optimization (PSO) é um algoritmo estocástico para otimização de funções não-lineares de alta dimensionalidade e com variáveis contínuas [3, 2]. O PSO foi introduzido por Kennedy e Eberhart em 1995 [4], e foi inspirado no comportamento social de organismos biológicos, como sendo um enxame de partículas que se movem pelo espaço e se comunicam para determinar uma direção de busca ideal. [3]. Cada partícula i do enxame S é representada por sua posição e sua velocidade. A posição é um vetor de n dimensões, cujos componentes representam os parâmetros da função objetivo. As partículas controlam a sua melhor posição *pbest* e a melhor posição global *gbest* ou seja a melhor solução conhecida dentro de sua vizinhança. No contexto do problema apresentado, uma partícula representa um fluxo de trabalho e suas tarefas, já a dimensão da partícula é igual ao número de tarefas no fluxo de trabalho e serve para indicar sua posição no espaço. Uma partícula movimenta-se num espaço limitado, o qual denominou-se alcance. O alcance é determinado pelo número de pools de threads disponíveis para executar a tarefa. A função de aptidão deve refletir os objetivos do problema de agendamento, pois ela é usada para determinar o quão boa uma determinada solução é. Logo, ela é minimizada e seu valor será o tempo total de execução *TTE* contido no agendamento A derivado da posição da partícula. A estratégia é definir um conjunto inicial de recursos que o algoritmo pode usar para explorar diferentes soluções e alcançar o agendamento. O tamanho desse conjunto será a restrição tratada $TR \leq \delta_r$, conforme a Equação 4. Considera-se um conjunto de recursos inicial $R_{inicial}$, composto por um *Pool* de cada tipo, para cada tarefa em P; onde P é o conjunto que contém o número máximo de tarefas que podem ser executadas em paralelo para um dado fluxo de trabalho. O algoritmo selecionará o número e o tipo apropriados de *Pools* para o motor de execução, dentro das opções contidas em $R_{inicial}$. Assim, a heterogeneidade dos recursos computacionais será considerada e o tamanho do espaço de busca, reduzido, além de permitir que o algoritmo mapeie todas as tarefas que podem ser executadas em paralelo. O Algoritmo 1 apresenta o pseudo-código para mapear a posição de uma partícula em um agendamento. O conjunto de recursos R para serem alocados e o conjunto de mapeamentos M de tarefas para recursos são inicializados sem elementos, ou seja, vazios, e tempo total de execução *TET* é inicializado com o valor zero. Na sequência, o algoritmo estima o tempo de execução de cada tarefa do fluxo de trabalho para todo o recurso $r_i \in R_{inicial}$. A representação é uma matriz em que as linhas representam as tarefas, as colunas representam os recursos e a entrada $TempoExec[i,j]$ corresponde ao tempo gasto para executar tarefa t_i no recurso r_j, calculado conforme a Equação 1. O próximo passo é o cálculo ou atribuição da matriz de tempo de transferência de dados, ou tempo de espera na fila de tarefas. Essa matriz é representada como uma matriz de adjacência ponderada, onde a entrada $TempoTransfer[i,j]$ contém o tempo que leva para transferir os dados de saída da tarefa t_i para a tarefa t_j e esse valor é zero sempre que $i = j$ ou não hádependência entre t_i e t_j. O algoritmo inicia determinando a posição da partícula e construindo o agendamento. Para isso, itera para toda i do *array* de posição *pos* e atualiza R e M. Primeiro, determina a tarefa e o recurso que está associado à coordenada atual e seu valor. A coordenada i corresponde à tarefa t_i e seu valor $pos[i]$ corresponde ao recurso $r_{pos[i]} \in R_{inicial}$. Encontrados os dois primeiros componentes, t_i e r_j, de uma tupla de mapeamento $m_{t_i}^{r_j}$, o algoritmo calcula os demais, o tempo de inicio $TIni_{t_i}$ e tempo de finalização $TFin_{t_i}$) da tarefa. Quando o algoritmo termina de processar, cada coordenada do vetor posição, R conterá todos os recursos necessários e os tempos de início e finalização. Além disso, o mapeamento completo das tarefas para os recursos estará em M e cada tarefa terá um recurso

atribuído a ela e o tempo estimado de início e o de término. Com essas informações o algoritmo pode calcular o *TTE* associado a solução atual, conforme Equação 2. Nesse ponto, o algoritmo calculou *R*, *M* e *TTE* e poderá construir e apresentar o agendamento associado para essa posição da partícula. O

Algorithm 1 Geração de agendamento

Entrada: Conjunto de fluxo de trabalho de tarefas T
Conjunto Inicial de recursos $R_{inicial}$
Um array $pos[|T|]$ representando a posição da partícula
Saída: Um agendamento A

1: $\mathtt{R} = \emptyset, \mathtt{M} = \emptyset, TTE = 0$ ▷ Inicializa componentes
2: Calcula $TempoExec[\,|T| \times |R_{inicial}|]$
3: Calcula $TempoTransf[\,|T| \times |T|]$
4: **for** i **do**$0|T| - 1$
5: $t_i = T[i], r_{pos[i]} = \mathtt{R}_{inicial}[pos[i]]$
6: **if** t_i não tem pai **then**
7: $TIni_{t_i} = TFin_{r_{pos[i]}}$
8: **else**
9: $TIni_{t_i} = (\max\{TFin_{t_{pai}} : t_{pai}[\in pais(t_i)\}, TFin_{r_{pos[i]}})$
10: **end if**
11: $exe = tempoExec\,[i]\,[pos\,[i]]$
12: **for** cada filha t_{filha} de t_i **do**
13: **if** t_{filha} é mapeada para um recurso diferente de $r_{pos[i]}$ **then**
14: $trasfer+ = TempoTransf[i][c]$
15: **end if**
16: **end for**
17: $TP_{t_i}^{r_{pos[j]}} = exe + transf$
18: $TFin_{t_i} = TP_{t_i}^{r_{pos[j]}} - TIni_{t_i}$
19: $m_{t_i}^{r_{pos[j]}} = (t_i, r_{pos[i]}, TIni_{t_i}, TFin_{t_i})$
20: $\mathtt{M} = \mathtt{M} \cup \{m_{t_i}^{r_{pos[i]}}\}$
21: **if** $r_{pos[i]} \notin \mathtt{R}$ **then**
22: $\mathtt{R} = \mathtt{R} \cup r_{pos[i]}$
23: **end if**
24: **end for**
25: Calcula *TTE* conforme Equação 3
26: $A = (R; M; TR; TTE)$

algoritmo do PSO e o do agendamento 1 são combinados para o agendamento próximo de ótimo. Utilizou-se o *TTE* como valor de aptidão nas etapas seguintes e introduziu-se o mecanismo de manipulação de restrição, garantindo que o $TR \leq \delta_r$.

IV. Conclução

Para acompanhar as tendências tecnológicas e otimizar os resultados de seus processos de negócios, as empresas integram suas aplicações utilizando plataformas de integração. Essas plataformas são ferramentas de software que devem oferecer um desempenho adequado com uma quantidade limitada dos recursos computacionais. O motor de execução das plataformas de integração é o componente responsável pela execução das soluções de integração, programas computacionais que integram as aplicações por meio de tarefas. O motor possui algoritmo de agendamento as tarefas e alocação para pools de threads que irão executá-las. Um algoritmo de agendamento de tarefas ineficiente, poderá levar a um aumento do tempo de execução, degradando o desempenho da solução de integração e aumentando os gastos da empresa, principalmente em ambientes que utilizam a computação em nuvem, a qual adota o sistema de cobrança proporcional ao tempo de utilização dos recursos computacionais.

Nesse artigo, utilizou-se a técnica de otimização metaheurística *Particle Swarm Optimization (PSO)* para modelar o problema do agendamento das tarefas de soluções de integração. Com o problema modelado no PSO, foi proposto um algoritmo que faz o agendamento das tarefas para os pools de threads, considerando a complexidade computacional da tarefa e diferentes capacidades computacionais dos pools de thread, buscando o menor tempo de execução, sem exceder um valor pré-definido para o total de recursos, promovendo uma melhoria no desempenho sem onerar o processo de negócio.

Como trabalho futuro, sugere-se experimentação do algoritmo, análise comparativa em relação a outras proposta de agendamento de tarefas e a validação dos dados de desempenho obtidos e dados reais.

Referências

[1] Ansu A Abraham, Gary M King, Daniel V Rosa, and Donald W Schmidt. Runtime capacity planning in a simultaneous multithreading (smt) environment, August 16 2016. US Patent 9,417,927.

[2] Mohammed R AlRashidi and Mohamed E El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13 (4):913–918, 2009.

[3] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120–127. IEEE, 2007.

[4] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[5] Rafael Z. Frantz, Sandro Sawicki, Fabricia Roos-Frantz, Rafael Corchuelo, Vitor Basto-Fernandes, and Inma Hernández. Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem. *XIX Jornada de Pesquisa*, pages 1–11, 2014.

[6] Ying Huang and Jen-Yao Chung. A web services-based framework for business integration solutions. *Electronic Commerce Research and Applications*, 2(1):15–26, 2003.

[7] Atul Vikas Lakra and Dharmendra Kumar Yadav. Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. *Procedia Computer Science*, 48:107–113, 2015.

[8] Cláudia O Ourique, Evaristo C Biscaia, and José Carlos Pinto. The use of particle swarm optimization for dynamical analysis in chemical processes. *Computers & Chemical Engineering*, 26 (12):1783–1793, 2002.

[9] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.

[10] Frans Van Den Bergh and Andries Petrus Engelbrecht. A study of particle swarm optimization particle trajectories. *Information sciences*, 176(8):937–971, 2006.

Uma Nova Proposta de Modelagem de Preços de Provedores de IaaS Aplicada à Integração de Aplicações Empresariais

Cássio L. M. Belusso*
Universidade Federal da Fronteira Sul
Cerro Largo, RS, Brasil
cassio.belusso@uffs.edu.br

***Resumo*—Uma alternativa para usuários reduzirem custos de manutenção e aquisição de infraestrutura computacional é a computação em nuvem. Os serviços de computação em nuvem são oferecidos por provedores e, dentre eles, está *Infrastructure-as-a-Service* (IaaS). Em IaaS, o usuário contrata instâncias de máquinas virtuais compostas por diferentes conf gurações de recursos computacionais a um determinado valor. Diante do alto custo de manutenção de infraestruturas locais, empresas têm migrado suas aplicações para a nuvem. O mesmo ocorre com as soluções de integração, as quais são peças-chave no processo de integração de aplicações empresariais. O desaf o enfrentado pelas empresas é escolher o melhor provedor/instância para migrar suas soluções de integração. Neste trabalho, busca-se auxiliar as empresas neste processo por meio de um estudo preliminar para elaboração de uma nova proposta de modelagem dos preços das instâncias dos provedores Amazon EC2, Google Compute Engine e Microsoft Windows Azure usando regressão linear múltipla.**

Palavras-chave: Computação em Nuvem; *Infrastructure-as-a-Service*; Modelo de Preços; Regressão Linear; Integração de Aplicações Empresariais.

I. Introdução

A computação em nuvem é um dos principais avanços realizados no campo da tecnologia da informação atualmente. Neste novo conceito de computação, o usuário pode fazer uso das mais diversas aplicações através da Internet, como se elas estivessem instaladas em seu próprio computador, pagando somente pelo que utiliza [7].

A diminuição de custos da construção de uma infraestrutura computacional própria a partir do fornecimento de uma infraestrutura centralizada e compartilhada é uma das vantagens que a computação em nuvem pode oferecer. Estudos baseados em simulações estimam que, em um intervalo de tempo de pouco mais de uma década, o custo total de implementar e manter um ambiente na nuvem pode ser até dois terços menor do que manter um centro de dados tradicional não virtualizado [1].

Os serviços de computação em nuvem são oferecidos por provedores e podem ser resumidos em três modalidades: *Software-as-a-Service* (SaaS), onde o usuário utiliza determinado *software* e paga pela utilização; *Platform-as-a-Service* (PaaS), onde o usuário dispõe de um ambiente para projetar, testar e implantar aplicações personalizadas; e *Infrastructure-as-a-Service* (IaaS), onde o usuário contrata máquinas virtuais e gerencia os recursos computacionais como desejar. Esta pesquisa foca em IaaS.

Os provedores de IaaS oferecem diferentes planos para contratação denominados instâncias e que variam de acordo com as necessidades do usuário. Os preços das instâncias baseiam-se na quantidade de recursos computacionais que as compõem. Porém, uma questão importante é compreender como estes preços são def nidos. Muitos fatores alteram o preço das instâncias, como o local onde a máquina virtual está hospedada ou o seu sistema operacional. Diante disso, os usuários enfrentam um dilema durante o processo de tomada de decisão, levando-os, muitas vezes, à decisões precipitadas.

Geralmente, grandes empresas possuem grandes demandas, pois possuem muitas aplicações executando simultaneamente. No contexto da Integração de Aplicações Empresariais (do inglês *Enterprise Application Integration* - EAI), a computação em nuvem proporciona uma infraestrutura computacional de alta capacidade a um baixo custo, onde as soluções de integração podem ser implantadas e executadas [2].

As soluções de integração são *softwares* fundamentais no processo de integração de aplicações empresariais, pois sua principal função é sincronizar dados entre aplicações distintas ou reutilizar suas funcionalidades. Elas são implantadas no ecossistema de *software* da empresa como uma nova aplicação, oferecendo aos seus usuários a possibilidade de interagir com diferentes tipos de aplicações. O grande número de requisições de entrada e saída e a necessidade de pouco armazenamento são algumas características que diferem uma solução de integração da maioria das aplicações convencionais.

Diante da falta de transparência dos provedores de IaaS na forma com que os preços de seus serviços são def nidos, faz-se necessária uma ferramenta capaz de auxiliar as empresas no processo de tomada de decisão do melhor provedor/instância para migrar suas soluções de integração para a nuvem. Para isso, é apresentado um estudo preliminar para elaboração de uma nova proposta de modelagem dos preços das instâncias praticados por três provedores de IaaS: Amazon EC2 [11] (no restante do texto somente Amazon), Google Compute Engine [10] (no restante do texto somente Google) e Microsoft Windows Azure [9] (no restante do texto somente Azure). Algumas hipóteses a serem adotadas neste novo modelo são discutidas e uma análise de preços de um pequeno grupo de

instâncias do Azure é realizada. Trata-se de um importante passo para tornar a política de preços dos provedores mais clara, de modo que as empresas possam reduzir os custos oriundos da implantação e execução de suas soluções de integração em infraestruturas na nuvem.

O restante deste artigo está organizado da seguinte forma: a Seção II apresenta a metodologia adotada na elaboração de uma nova proposta de modelagem de preços das instâncias; na Seção III são apresentadas informações preliminares que justif cam a escolha do modelo de regressão; na Seção IV são apresentadas as conclusões e algumas perspectivas para trabalhos futuros.

II. Metodologia

Algumas pesquisas buscaram, de diferentes formas, tornar a política de preços adotadas por provedores de IaaS mais transparente [4, 6, 1, 3, 5, 8]. Apesar disso, nenhuma delas teve foco na EAI. Diante deste cenário, a necessidade de um mecanismo capaz de auxiliar as empresas no processo de tomada de decisão para a migração de soluções de integração para a nuvem é um passo importante neste contexto.

Nesta seção, é apresentado o esboço de uma nova proposta de modelagem de preços adotadas por provedores de IaaS com foco na migração de soluções de integração para a nuvem. Devido a essa particularidade, algumas hipóteses precisam ser discutidas e estabelecidas nesta primeira etapa, considerando objetivos próprios e público-alvo. Diante disso, destacam-se, inicialmente, as seguintes hipóteses simplif cadoras:

- Provedores: Amazon, Google e Azure;
- Sistema Operacional da Máquina Virtual: Linux e Windows;
- Tipo de Instância: *High-memory* e *High-CPU*;
- Localização Geográf ca do Provedor: Limitada a cinco;
- Modelo de Cobrança: *On-demand* e *Reserved*;
- Período de Utilização: *Peak hours* e *Off-peak hours*.

O objetivo assumido nesta proposta de modelagem de preços é mapear as instâncias para possibilitar a identif cação da instância com o menor preço, mas com qualidade de serviço capaz de executar a demanda computacional pré-determinada. Isso decorre do pressuposto de que nem sempre a opção pelo menor preço pode ser a melhor escolha. Nestes casos, é importante f car atento aos requisitos exigidos pela aplicação.

Diante dos inúmeros provedores no mercado da computação em nuvem, optou-se pelo Amazon, Google e Azure. A escolha pelo Amazon foi baseada no grande número de instâncias oferecidas e de mais modalidades de contratação; Google foi escolhido pela popularidade e pela simplicidade na consulta dos preços; a escolha do Azure deu-se pela grande quantidade de localizações geográf cas com máquinas virtuais hospedadas. Apesar das hipóteses abordarem os três provedores, uma análise de preços mais detalhada foi realizada apenas para o Azure, cuja escolha se deve apenas por ter sido o único até o momento a ter os preços estudados de forma mais detalhada.

Quanto ao tipo de instância, as *High-memory* possuem grandes quantidades de memória e as *High-CPU* têm uma grande capacidade de processamento. Quanto ao período de utilização, caso o usuário utilize a máquina virtual em horários considerados de pico (*peak hours*), o custo será maior do que nos demais (*off-peak hours*).

O processo de coleta de dados é um passo importante no processo de modelagem. Para isso, foram simuladas contratações de instâncias do Azure em todas as suas localizações geográf cas, bem como as informações referentes aos sistemas operacionais Linux e Windows. O modelo de cobrança adotado foi o *On-demand*, pois ele é o mais atrativo para a maioria dos perf s de usuário. Neste modelo, não existe compromisso a longo prazo e o usuário pode cancelar o serviço a qualquer momento. No modelo *Reserved*, o usuário contrata uma instância por um determinado período de tempo.

Diante da grande quantidade de dados obtidos, retirou-se uma pequena amostra composta por um grupo de cinco instâncias, as quais foram escolhidas considerando suas disponibilidades em mais localizações geográf cas. A opção escolhida foi a região Leste dos EUA, pois é um dos locais mais baratos e disponibiliza uma maior variabilidade de instâncias. O sistema operacional adotado foi o Windows, pois é o mais utilizado atualmente, e a camada de preços foi a Padrão, pois ela oferece maior f exibilidade. As instâncias do Azure são formadas pelos recursos computacionais CPU, memória e armazenamento. A cobrança é feita por hora de utilização e os dados são referentes ao mês de outubro de 2016.

As variáveis do modelo dividem-se em variáveis quantitativas, compostas pelos requisitos de *hardware* (por exemplo, CPU e memória), e variáveis qualitativas, compostas pelos requisitos de *software* (por exemplo, sistema operacional) e pelas demais variáveis (por exemplo, localização geográf ca). A representação das variáveis qualitativas do modelo são feitas por meio de variáveis Dummy, em que uma determinada categoria pode assumir o valor 0 ou 1 [12]. As variáveis Dummy são def nidas e inseridas previamente no modelo e representam as variáveis que não podem ser quantif cadas.

III. Modelagem Matemática

O principal objetivo da retirada de uma amostra de dados foi analisar o preço das instâncias mediante variação dos principais recursos computacionais. Para isso, nesta seção, são utilizados gráf cos de dispersão para verif car a forma de distribuição dos dados. As marcações presentes nos gráf cos representam as cinco instâncias do Azure escolhidas, da menor para a maior.

Na Figura 1 são apresentados gráf cos de dispersão da quantidade de CPU e memória *versus* a taxa temporal de utilização, computada em horas. Pode-se perceber um comportamento linear para ambos os recursos computacionais em praticamente todas as instâncias, exceto na menor. Além disso, especif camente no gráf co da memória, nota-se uma leve mudança no coef ciente angular da reta na segunda instância.

Para melhor vizualização, na Figura 2 é mostrado separadamente o gráf co do armazenamento *versus* a taxa de utilização. Novamente tem-se a presença do comportamento linear para este recurso e uma leve mudança no coef ciente angular da reta é detectada, neste caso, na terceira instância.

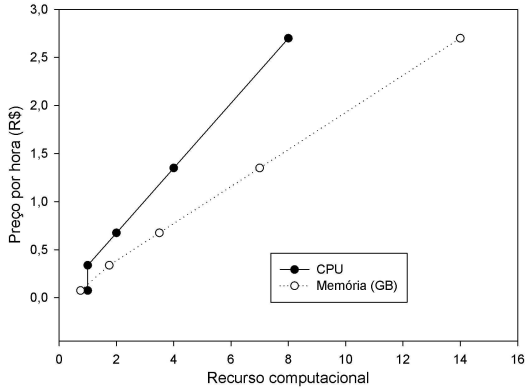

Figura 1. Gráfico de dispersão do recurso computacional *versus* preço.

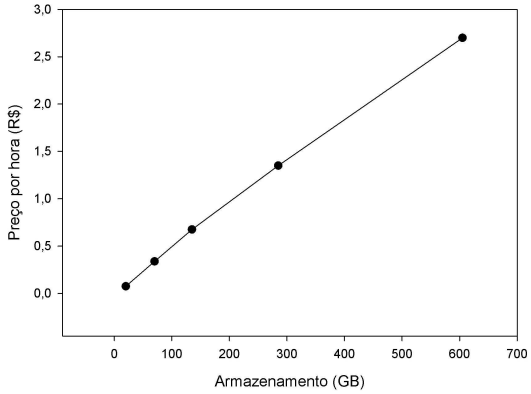

Figura 2. Gráfico de dispersão do armazenamento *versus* preço.

Mediante o comportamento linear dos dados, optou-se pela regressão linear múltipla para estimar o preço das instâncias por haver mais de uma variável independente. Com o método de regressão é possível estimar o custo individual das características que influenciam no preço final das instâncias por meio do cálculo dos coeficientes referentes a cada uma das variáveis do modelo. O modelo de regressão linear múltipla adaptado para esta proposta encontra-se descrito pela Equação 1. Os coeficientes a_i referentes às variáveis quantitativas representam o preço unitário de cada um dos recursos, ou seja, o custo por cada unidade de memória, de CPU e de armazenamento.

$$C_i = a_0 + a_1X_{i1} + a_2X_{i2} + ... + a_nX_{in} + \varepsilon_i, i = 1, ..., n \quad (1)$$

Onde:

- i refere-se à i-ésima instância, de qualquer um dos provedores escolhidos;
- n é o número de variáveis do modelo;
- C_i é o preço final da instância i;
- a_i são os coeficientes da regressão a serem calculados, ou seja, a fatia de participação da característica X_{in} no preço final da instância i;
- X_{in} são as variáveis independentes do modelo, no caso cada uma das características que influenciam no preço final da instância i;
- ε_i é o erro residual da regressão para cada instância i.

IV. Conclusão

Neste trabalho foi apresentado um estudo preliminar para a elaboração de uma nova proposta de modelagem dos preços das instâncias praticados por provedores de computação em nuvem a nível de IaaS utilizando um modelo de regressão linear múltipla. Algumas hipóteses para a criação desta nova metodologia foram discutidas considerando os provedores Amazon, Google e Azure. Gráficos de dispersão foram utilizados para justificar a escolha do modelo de regressão por meio de uma análise dos preços de um pequeno grupo de instâncias do Azure.

Esta é uma importante etapa no processo de criação de uma ferramenta capaz de auxiliar as empresas no processo de tomada de decisão durante a migração de suas soluções de integração para a nuvem. A revisão da literatura mostrou que nenhuma proposta encontra-se direcionada à EAI. A ausência de um mecanismo que possa oferecer às empresas uma forma de comparar os serviços oferecidos pelos provedores torna esta nova abordagem promissora, pois pode tornar a tomada de decisão uma tarefa mais simples e menos onerosa.

Para trabalhos futuros, espera-se realizar a análise de preços para os dois provedores restantes, Amazon e Google, e implementar um modelo de regressão para cada um. Por se tratar de um método de estimativa, pretende-se melhorar o nível de significância acrescentando novas variáveis ao modelo.

Referências

[1] Se-Hak Chun and Byong-Sam Choi. Service models and pricing schemes for cloud computing. *Cluster Computing*, 17(2):529–535, 2014.

[2] Inma Hernández, Sandro Sawicki, Fabricia Roos-Frantz, and Rafael Z. Frantz. Cloud configuration modelling: a literature review from an application integration deployment perspective. *Procedia Computer Science*, 64:977–983, 2015.

[3] Jianhui Huang, Robert J. Kauffman, and Dan Ma. Pricing strategy for cloud computing: A damaged services perspective. *Decision Support Systems*, 78:80–92, 2015.

[4] Siham El Kihal, Christian Schlereth, and Bernd Skiera. Price comparasion for infrastructure-as-a-service. In *Proceedings of the ECIS12*, 2012.

[5] Michael Menzel and Rajiv Ranjan. Cloudgenius: Decision support for web server cloud migration. In *Proceedings of the WWW12*, pages 979–988, 2012.

[6] Persefoni Mitropoulou, Evangelia Filiopoulou, Christos Michalakelis, and Mara Nikolaidou. Pricing cloud IaaS services based on a hedonic price index. *Computing*, 98(11):1075–1089, 2016.

[7] M. K. Mohan Murthy, H. A. Sanjay, and Ashwini Janagal Padmanabha. Pricing models and pricing schemes of IaaS providers: a comparison study. In *Proceedings of the ICACCI12*, pages 143–147, 2012.

[8] Parnia Samimi and Ahmed Patel. Review of pricing models for grid & cloud computing. In *Proceedings of the ISCI 2011*, pages 634–639, 2011.

[9] Microsoft Azure. https://azure.com/. Acesso em: 24/10/2016.

[10] Google Cloud Plataform. https://cloud.google.com/. Acesso em: 25/10/2016.

[11] Amazon Web Services. https://aws.amazon.com/. Acesso em: 28/10/2016.

[12] Thomas H. Wonnacott and Ronald J. Wonnacott. *Introductory Statistics for Business and Economics*. John Wiley & Sons, 1990.

Automatização de Modelos de Simulação da Plataforma de Integração Guaraná

Igor G. Haugg*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brazil
igor-haugg@hotmail.com

***Resumo*—Atualmente existe uma grande quantidade de empresas que necessitam realizar integração entre suas aplicações. Isso ocorre porque as aplicações utilizadas são normalmente desenvolvidas por diferentes empresas e seguindo distintas tecnologias. Soluções de integração podem ser desenvolvidas através de diferentes plataformas. A plataforma Guaraná fornece componentes para modelar e implementar as soluções, além disso, pode ser utilizada na web através da ferramenta Guaraná Cloud. Nesta ferramenta o processo de desenvolvimento pode ser abstraído em: modelagem, implantação e monitoramento. Na modelagem cria-se o modelo conceitual da solução de integração, no qual será projetado o fluxo de integração desejado. Na implantação os engenheiros de software precisam definir a configuração do motor de execução, a qual é fundamental para garantir um bom desempenho e atingir os requisitos de qualidade de serviço que o usuário deseja. Monitoramento permite acompanhar a execução da solução de integração. Este trabalho tem como objetivo proporcionar uma ferramenta para auxiliar os engenheiros de software a encontrar a configuração ideal que precisa ser realizada no motor de execução para que o mesmo atenda aos requisitos de qualidade de serviço de uma solução de integração. Palavras-chave: Integração de Aplicações Empresariais, Simulação, Motor de Execução.**

I. Introdução

As empresas utilizam diversas aplicações, que geralmente são desenvolvidas na própria empresa ou adquiridas por terceiros. Esse conjunto de aplicações é chamado de ecossistema de software. Normalmente estas aplicações são adquiridas ao longo do tempo, e assim, o ecossistema de software torna-se heterogêneo, contendo aplicações que foram desenvolvidas em diferentes linguagens, diferentes sistemas operacionais, e que geralmente não foram projetadas para trabalharem de forma conjunta. A área de integração de aplicações empresariais (EAI) busca oferecer metodologias, técnicas e ferramentas para fazer com que diferentes aplicações, que não foram desenvolvidas com o propósito de trabalhar em conjunto, possam colaborar [4].

Soluções de integração podem ser desenvolvidas através de diversas plataformas, como por exemplo a plataforma Guaraná [2], a qual fornece componentes para modelar e implementar as soluções, tais como: linguagem de domínio específico (DSL), kit de ferramentas de desenvolvimento, ferramenta de monitoramento e motor de execução. DSL é uma linguagem que possibilita descrever os modelos conceituais das soluções de integração. O kit de desenvolvimento permite transformar os modelos conceituais em código executável. A ferramenta de monitoramento é utilizada analisar o estado da solução durante a sua execução. O motor é responsável pela execução da solução de integração, ele interpreta a solução implementada e proporciona todo o suporte necessário para a sua execução, além disso, tem impacto direto no desempenho da solução de integração. A sua arquitetura é organizada em torno de uma fila FIFO (primeiro elemento que entra é o primeiro a ser executado), onde há vários produtores que inserem elementos e servidores que realizam o atendimento.

A plataforma Guaraná pode ser utilizada na web através da ferramenta Guaraná Cloud, a qual permite o desenvolvimento e implantação de uma solução de integração. Nesta ferramenta, o processo de desenvolvimento de uma solução de integração pode ser abstraído nas etapas: modelagem, implantação e monitoramento. Na etapa de modelagem cria-se o modelo conceitual da solução, onde é projetado o fluxo de integração desejado. Na implantação os engenheiros de software precisam definir a configuração do motor de execução. Esta configuração é fundamental para garantir bom desempenho e também atingir os requisitos de qualidade de serviço que o usuário deseja, como por exemplo, o número de mensagens que está solução deve processar por unidade de tempo. Na fase de monitoramento a ferramenta permite acompanhar a execução da solução de integração, em tempo real é possível visualizar o tamanho da fila, quantidade de mensagens processadas, dentre outras informações.

Neste trabalho será descrito uma proposta de automatização da geração de modelos de simulação, com o objetivo de proporcionar uma ferramenta para auxiliar os engenheiros de software a encontrar a configuração ideal que precisa ser realizada no motor de execução para que esse atenda a requisitos de qualidade de serviço.

II. Proposta

A proposta de automatização de modelos de simulação tem como objetivo proporcionar uma forma de gerar automaticamente modelos de simulação para soluções de integração desenvolvidos na ferramenta Guaraná Cloud. Esta proposta está dividida em cinco etapas.A primeira etapa é o desenvolvimento do modelo conceitual da solução de integração na ferramenta

* Bolsista PROSUP/CAPES

Guaraná Cloud. A segunda etapa será realizada após a finalização do modelo conceitual, onde através da ferramenta Guaraná Cloud é possível gerar um modelo XML (*eXtensible Markup Language*) da solução desenvolvida. A geração do XML é realizada de forma automática pela plataforma, ou seja, o usuário não precisa ter conhecimento de como desenvolver um XML. Na terceira etapa será utilizada a ferramenta de automatização, a qual recebe como entrada um arquivo XML, este arquivo pode ser referente a qualquer solução de integração, desde que tenha sido desenvolvido na plataforma Guaraná. Nesta etapa, é necessário que o usuário informe algumas configurações para o modelo, como: número de *threads*, taxa de entrada de mensagens e o tempo que cada tarefa leva para ser executada. Na quarta etapa, é realizada a geração do modelo de simulação específico para o modelo conceitual recebido. Esses modelos são focados para simulação no software Simulink Matlab, pois esta é uma ferramenta que permite modelar, simular e analisar sistemas dinâmicos. Além disso, possui a biblioteca *SimEvents*, que permite o desenvolvimento de modelos para sistemas baseados em filas. A quinta etapa é a realização da simulação computacional, utilizando o modelo de simulação gerado pela ferramenta de automatização.

III. Caso de Estudo

Para a demonstração da ferramenta de automatização foi utilizado o caso de estudo Café [3]. Este caso de estudo descreve como os pedidos dos clientes são processados em uma cafeteria. O processo começa por um cliente solicitando um pedido para o caixa, que registra o pedido no sistema e o adiciona a uma fila de pedidos. Um pedido pode incluir entradas para bebidas quentes e frias, que são preparadas por diferentes baristas. Quando todas as bebidas que correspondem à mesma ordem foram preparadas, elas são entregues pelo garçom [1].

A. Modelo conceitual de integração

Para resolver este problema a solução de integração deve poder receber pedidos da fila de pedidos, enviar solicitações aos baristas para preparar as devidas bebidas e notificar o garçom quando uma encomenda estiver concluída. A Figura 1 apresenta uma solução utilizando a tecnologia Guaraná, através da ferramenta web Guaraná Cloud [1].

A solução de integração inicia na porta de entrada P1, que aguarda por novos pedidos. Cada pedido resulta em uma mensagem com as bebidas a serem preparadas, as mensagens geradas são adicionadas ao *slot* S1. A Tarefa T1 tem como funcionalidade, dividir cada mensagem em várias outras mensagens, para que seja possível enviar o pedido para o barista correto, ou seja, a parte da mensagem que consta o pedido de uma bebida quente é enviado para o barista de bebidas quentes e da mesma forma com as bebidas frias. A partir deste momento, as mensagens são encaminhadas para a tarefa T2, esta tarefa envia as mensagens para o destino correto. A Tarefa T3 replica as mensagens para a aplicação *Barista Cold Drinks*, de modo que uma cópia possa ser enviada para o barista e outra cópia para a tarefa T5, a qual correlaciona a resposta do barista com a cópia em espera. A tarefa T6 enriquece a cópia em espera com a informação devolvida pela aplicação *Barista Cold Drinks*. A Tarefa T4 transforma as mensagens para o formato necessário para que a aplicação *Barista Cold Drinks* possa entender. As mensagens que forem enviadas para a aplicação *Barista Hot Drinks* se comportam da mesma maneira. Após a preparação da bebida, as mensagens dos baristas são reunidas em um único *slot* S2 pela tarefa *merger* T7. Em seguida, as bebidas preparadas são retiradas deste *slot* e reagrupadas para uma única mensagem novamente, de modo que a porta de saída P6 escreve a mensagem resultante para a aplicação *Waiter* [1].

B. Modelo de Simulação

Na Figura 2 pode ser visualizado o modelo de simulação gerado a partir da ferramenta de automatização. O modelo é dividido em um conjunto de entidades, uma fila, um servidor e geradores de gráficos. Observa-se que para cada caso de estudo diferente, o modelo de simulação precisa ser ajustado somente na parte das tarefas (*tasks*), pois a mesma representa as tarefas da solução de integração do caso de estudo.

As entidades possuem um valor que determina o intervalo entre duas gerações de entidades, ou seja, o bloco *Entity* gera uma entidade e este valor determina quanto tempo levará para que uma nova entidade seja gerada. No modelo de simulação, este valor é calculado a partir da taxa de entrada informada na configuração da ferramenta de automatização. Além disso, nos blocos *Entity* foram adicionados dois atribuídos: tipo e tempo de serviço. O tipo é utilizado para diferenciar entre uma entidade e outra, também permite que as entidades sejam identificadas em outros blocos do sistema de simulação. O atributo tempo representa o tempo de serviço que os servidores levarão para atender a entidade, este valor também é informado durante a configuração da ferramenta, cada tarefa pode ter valores diferentes, conforme Figura 3.

As entidades geradas nos blocos *Entity* são encaminhadas para o bloco *Entity Input Switch*, este bloco tem como função encaminhar as entidades para o bloco *Work Queue*, o qual representa a fila de tarefas. Na fila é possível a geração de diversos tipos de gráficos, os quais representam estatísticas do sistema. Neste caso foram selecionados os gráficos de tamanho da fila e o tempo médio de espera na fila. A fila segue a disciplina FIFO (*First In First Out*), pois a arquitetura do motor de execução da plataforma Guaraná está organizada em torno de uma fila FIFO.

O bloco *Threads/Workers* representa os servidores, e no contexto de integração de aplicações, representa as *threads* disponíveis para a execução da solução de integração. Após a execução das entidades, este bloco encaminha as entidades para a sua finalização, que ocorre no bloco *Entity Terminator*.

Na Figura 3 é possível visualizar a interface gráfica da ferramenta de automatização, onde já está carregado o XML referente ao caso de estudo Café. Também é possível visualizar as configurações necessárias, como: número de *threads* e tempo de execução para cada tarefa.

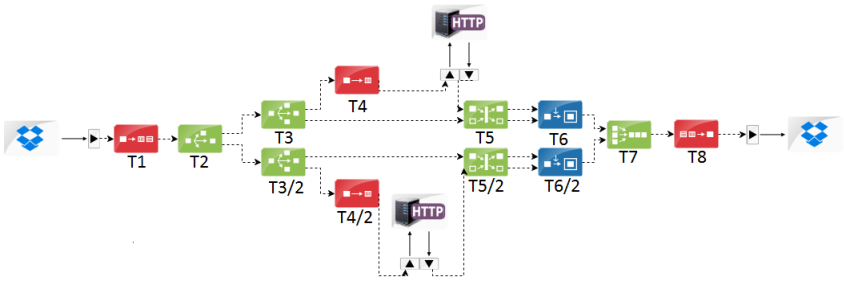

Figura 1. Modelo Conceitual da Solução de Integração Café.

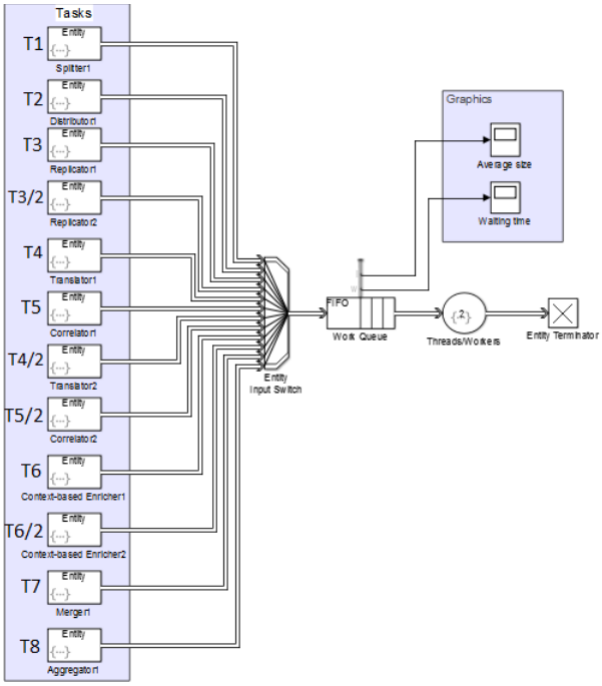

Figura 2. Modelo de Simulação da Solução de Integração Café.

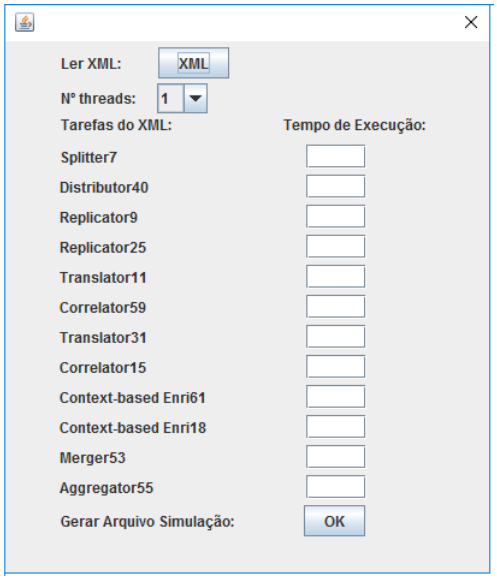

Figura 3. Ferramenta de Automatização.

IV. Conclusão

Com o desenvolvimento deste trabalho foi possível concluir que a utilização de uma ferramenta para automatizar o processo de desenvolvimento de modelos de simulação possui grandes benefícios, com a ferramenta é possível rapidamente gerar um modelo e realizar a simulação para conhecer o comportamento da solução de integração desenvolvida, antes de executar a solução de integração em um cenário real. Além disso, é possível construir cenários diferentes para a solução desenvolvida, e realizar experimentos nestes cenários. Porém, como a ferramenta foi desenvolvida baseada na arquitetura do motor de execução, somente poderá simular modelos desenvolvidos para esta arquitetura.

Como trabalho futuro, pretende-se realizar uma alteração na forma como é gerado o tempo de execução para cada tarefa no modelo de simulação, de forma com que não seja necessário que o usuário informe os tempos de execução para cada tarefa. Desta forma, será de responsabilidade do sistema, informar quanto tempo cada tarefa leva para ser executada, baseando-se em experimentos que serão realizados em casos reais na plataforma de integração Guaraná Cloud. Além disso, pretende-se realizar uma comparação com outras disciplinas de fila, para descobrir se há uma forma de executar mais mensagens em um menor tempo de execução.

Referências

[1] Rafael Z Frantz. *Enterprise Application Integration An Easy-to-Maintain Model-Driven Engineering Approach*. PhD thesis, Doctoral Dissertation. University of Seville, 2012.

[2] Rafael Z Frantz, Rafael Corchuelo, and Fabricia Roos-Frantz. On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111:89–104, 2016.

[3] Gregor Hohpe. Your coffee shop doesn't use two-phase commit [asynchronous messaging architecture]. *IEEE software*, 22(2):64–66, 2005.

[4] Gregor Hohpe and Bobby Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[5] David S Linthicum. *Enterprise application integration*. Addison-Wesley Professional, 2000.

[6] David G Messerschmitt, Clemens Szyperski, et al. Software ecosystem: understanding an indispensable technology and industry. *MIT Press Books*, 1, 2005.

Análise do Desempenho da API Executor, das Implementações Zulu JVM e Oracle JVM, nos Sistemas Operacionais Windows e Linux

Eldair F. Dornelles*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brasil
{eldair.dornelles}@gmail.com

***Resumo*—O uso de *threads* possibilitam a execução simultânea de tarefas, aumentando assim o desempenho dos *softwares*, minimizando o tempo de resposta e aumentando o atendimento às requisições. Neste cenário, é destacado o uso da programação multithread, a qual permite que um mesmo processo possa ser dividido e executado em diferentes núcleos de um processador. No entanto, a criação e gerenciamento de threads pode se tornar uma tarefa complexa. Para auxiliar na criação e gerenciamento das threads são disponibilizadas algumas bibliotecas. Para essa finalidade, na linguagem de programação Java é disponibilizado a API Executor. A API Executor, é uma interface de programação, a qual oferece uma série de recursos que auxiliam na criação e gerenciamento das threads. Para o desenvolvimento das aplicações Java, é necessário possuir a Java Virtual Machine (JVM) e o Java Development Kit (JDK) devidamente configurado. Atualmente existe um grande número de implementações JVMs, desenvolvidos por diferentes empresas. Assim, esse artigo realiza um estudo comparativo sobre o desempenho da API Executor, das implementações Zulu JVM e Oracle JVM, nos sistemas operacionais Windows e Linux, considerando as variáveis tempo de uso, tempo de CPU e tempo de sistema. Foi observado similaridades no desempenho das JVMs considerando os sistemas operacionais separadamente. Destacando ainda que as implementações da API Executor, de ambas as JVMs estudadas, não apresentam diferenças significativas quanto aos tempos medidos neste estudo.**

Palavras-chave: Multithread; Executor; Máquina Virtual Java; Desempenho.

I. Introdução

A utilização de *threads* visam possibilitar a execução simultânea de tarefas [5, 13]. Deste modo, aumentam o desempenho dos *softwares*, atendendo maior demanda de tarefas em menor tempo. Embora o uso de *threads* ofereçam vantagens ao desempenho dos *softwares*, elas podem tornar o desenvolvimento complexo, pois devem atender tanto a atributos de qualidade quanto aos parâmetros de desempenho previamente configurados. Tais *softwares* devem oferecer um desempenho adequado, de modo que minimizem o tempo de resposta e aumentem o atendimento às requisições. Para tanto, é necessário, além de conhecimento à codificação e manipulação de *threads*, atentar também, sobre as tecnologia disponibilizadas pelas diferentes plataformas de desenvolvimento de *softwares* [8]. Um estudo comparativo do desempenho entre diferentes plataformas de desenvolvimento, pode representar uma alternativa eficiente à identificação da melhor plataforma ao desenvolvimento de um software. Neste contexto, o objetivo deste estudo é analisar o desempenho da JVM Zulu, desenvolvida pela empresa Azul System [3] e a HotSpot JVM, desenvolvida pela empresa Oracle [1]. O restante deste artigo tem sua estrutura organizada como segue. A Seção 2 apresenta informações gerais sobre Máquina virtual Java. A seção 3 relata os experimentos realizados a respeito do desempenho das JVMs analisadas. Por fim, a Seção 4 elenca as principais conclusões.

* Bolsista PROSUP/CAPES

II. Background

A. Máquina Virtual Java

Java é uma completa plataforma de desenvolvimento e execução, composta por três elementos: a JVM, um conjunto de APIs e a linguagem de programação Java. A JVM é uma aplicação que abstrai tanto a camada de hardware como a de comunicação com o sistema operacional. Isso significa que, ao utilizar o conceito de máquina virtual, o compilador Java gera um programa executável para uma máquina genérica, virtual e não física, que possui suas próprias instruções de máquina e suas APIs. Sua função é executar as instruções de máquina genéricas no sistema operacional e no hardware específico sob o qual estiver rodando. Assim, é necessário instalar a JVM adequada para o sistema operacional e hardware utilizados.

Uma JVM é uma especificação, ou seja, normas para o desenvolvimento do software, de forma que, um fabricante ou mesmo um desenvolvedor pode escrever sua própria máquina virtual para o Java. Como exemplos dessas JVMs cita-se as utilizadas nesse estudo: HotSpot Java da Oracle [1] e Zulu da Azul Systems [3]. Cada fabricante tenta melhorar sua própria implementação, utilizando diferentes estratégias, tais como: aperfeiçoamento do compilador, do gerenciamento de memória, do escalonamento das threads, promovendo assim a competitividade do mercado e oferecendo mais opções para os desenvolvedores [12].

O núcleo da JVM é seu motor de execução, cujo comportamento é definido por um conjunto de instruções. Cada thread do programa é uma instância do motor de execução, que executa `bytecodes` ou métodos nativos, esses últimos

são códigos escritos em outra linguagem e compilados para código de máquina nativo de um particular processador. Além das threads do programa, a JVM pode usar outras threads, transparentes ao programa, como por exemplo, as threads coletoras de lixo (`garbage collection`), as quais não precisam ser instâncias do motor de execução [9].

III. Experimento

Nesta seção são apresentadas as configurações do computador e dos sistemas operacionais utilizados. É descrito a forma como o experimento foi estruturado e executado. Além disso, é detalhado as variáveis analisadas bem como a analise estatística definida.

A. Configuração do Ambiente

O experimento foi realizado em um computador com processador Intel® Celeron® 1.836 GHz, com 4 núcleos físicos e 4 processadores lógicos, 4G de memória RAM, 64 bits e no sistema operacional Windows e LINUX Fedora. Os critérios utilizados na seleção das JVMs foram:
(i) suportar o sistema operacional Windows e LINUX;
(ii) suportar processador com base x64 e
(iii) possuir versão atualizada.
Atendendo a estes requisitos, foram selecionadas as JVMs:
(i) Java HotSpot 25.101-b13-jdk.8.0.101 da Oracle;
(ii) Zulu 8.17.0.3-jdk8.0.102 da Azul Systems.

B. Descrição do Experimento

O estudo consiste em medir o tempo de CPU, o tempo do sistema e tempo de uso consumidos na execução de um mesmo algoritmo em cada uma das JVMs, em ambos os sistemas operacionais e comparar os resultados. O tempo de CPU não conta Entrada/Saída (E/S) ou tempo gasto executando outros programas e esse tempo pode ser dividido em tempo de sistema e tempo de uso. O tempo de sistema é o tempo consumido executando comandos do próprios do sistema e mais algum *overhead*, como por exemplo, o tempo da troca de processo a ser executado. Já o tempo de uso é tempo gasto executando as linhas de código que estão no programa implementado para o experimento, desconsiderando chamadas e tratamento por parte do sistema operacional [11].

Foi implementado uma classe Java chamada `Experiment`, a qual extende a classe `Thread`. Nessa classe foi criado um pool com duas threads, através do método `newFixedThrheadPool()`. A classe `Experiment` contabiliza apenas o custo computacional, não trata interrupções de E/S. `Experiment` cria dez tarefas, onde cada tarefa computa os números primos entre 1 a 10.000; faz medições dos tempos; e registra os resultados das medições que utilizados nas análises estatísticas. O algoritmo dos números primos foi implementado na classe `Prime`. As medições dos tempos foram implementadas na classe `ThreadMonitor` [6]. A classe `Start` representa a classe principal, com método *main*, a partir da qual se inicia o experimento criando uma instância da classe `Experiment`, que é executada vinte e cinco vezes, coletando os tempos do experimento. Na Figura 1 é apresentada a forma de execução do experimento.

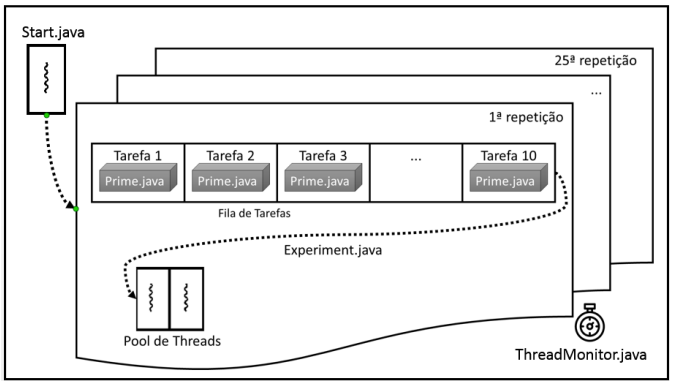

Figura 1. Execução do experimento.

C. Análise Estatística

Na análise dos dados, foi realizado a média aritmética dos valores observados para cada variável, considerando as 25 repetições. Os valores das médias obtidos para tempo de CPU, tempo de sistema e tempo de uso, foram calculados separadamente para cada JVM e sistema operacional.

D. Resultado e discussões

No gráfico da Figura 2, estão apresentados os valores obtidos na execução do experimento. É observado que as JVMs apresentam desempenhos similares entre si. Sendo que a maior diferença de desempenho observada, foi na variável tempo de uso, no sistema Linux, onde a JVM Zulu foi 0,4 segundos mais rápida que a JVM da Oracle. Nas variáveis tempo de CPU e tempo de sistema, as diferença de desempenho ficaram abaixo de 0,055 segundos, em ambos os sistemas operacionais. Destaca-se que na comparação de desempenho das JVMs entre os sistemas operacionais, as variáveis tempo de uso e tempo de cpu, obtiveram melhor desempenho no sistema Linux Fedora, já a variável tempo de sistema apresentou desempenho superior no sistema operacional Windows.

As diferenças observadas entre as medições dos tempos entre as JVMs em cada sistema operacional, separadamente, podem ser atribuídas as oscilações do ambiente na execução de outros programas ou na implementação interna das JVMs, como por exemplo, otimização feita pelo compilador, sendo esses, fatores não relacionados com o gerenciamento das threads realizada com as classes da API Executor do Java.

IV. Conclusão

O artigo apresentou uma comparação entre duas JVMs, quanto ao desempenho da API Executor, considerando os sistemas operacionais Windows e Linux Fedora. A análise comparativa foi realizada entre a HotSpot JVM da Oracle [2] e Zulu JVM da Azul Systems [3]. O estudo do desempenho foi realizado por meio de experimentos e comparação de médias. Na análise estatística foi observado similaridades no desempenho das JVMs considerando os sistemas operacionais

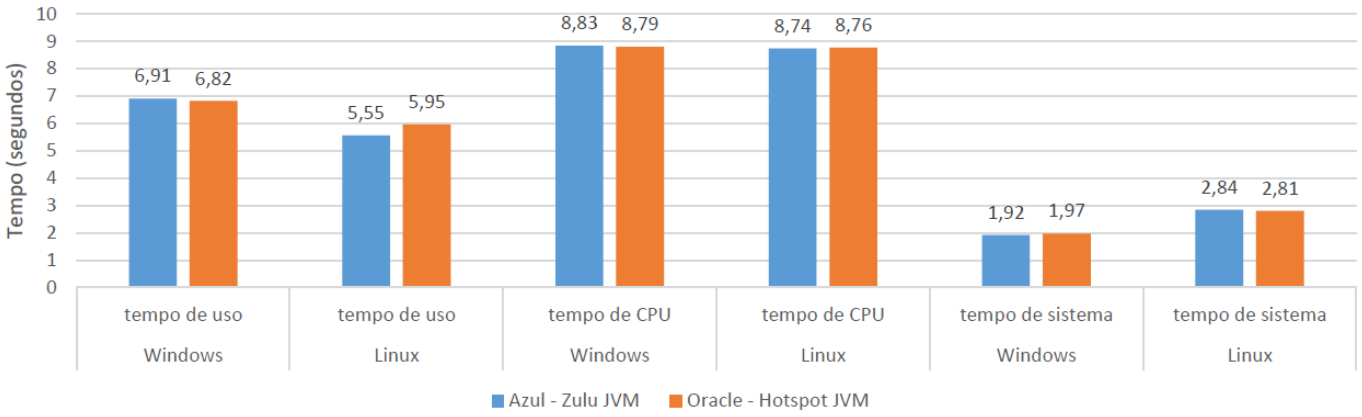

Figura 2. Médias dos tempos (tempo de uso, tempo de CPU e tempo de sistema) de ambas as JVMs e sistemas operacionais

separadamente. Assim, pode-se afirmar que as implementações da API Executor, de ambas as JVMs estudadas, não apresentam diferenças significativas quanto aos tempos medidos neste estudo.

Referências

[1] Hotspot. URL `http://www.oracle.com/technetwork/systems/vmoptions-jsp-140102.html`.

[2] Openjdk. URL `http://openjdk.java.net/`.

[3] Zulu. URL `https://www.azul.com/products/zulu/`.

[4] List of java virtual machines. URL `https://en.wikipedia.org/wiki/List_of_Java_virtual_machines`.

[5] Paul Dietel. *Java how to program*. PHI, 2009.

[6] Rafael Z. Frantz. *Enterprise application integration: an easy-to-maintain model-driven engineering approach*. PhD thesis, Universidad de Sevilla, 2012.

[7] Andy Georges, Dries Buytaert, and Lieven Eeckhout. Statistically rigorous java performance evaluation. *ACM SIGPLAN Notices*, 42(10):57–76, 2007.

[8] Gleydson de Azevedo Ferreira Lima. Análise de desempenho de sistemas distribuídos de grande porte na plataforma java. Master's thesis, Universidade Federal do Rio Grande do Norte, 2007.

[9] Francis Rangel and Anderson Faustino Silva. A máquina virtual java e a otimização inline: Um estudo de caso. *Revista Tecnológica*, 21(1):103–118, 2012.

[10] Robert G Sargent. A new statistical procedure for validation of simulation and stochastic models. 2010. URL `http://surface.syr.edu/eecs/175/`.

[11] Gabriel P Silva. Avaliação de um ambiente UNIX com múltiplos processadores. In *Anais do II Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo*, volume 2, page 11.8.6.1, 1988.

[12] Paulo Silveira, Guilherme Silveira, Sérgio Lopes, Guilherme Moreira, Nico Steppat, and Fábio Kung. *Introdução à arquitetura e design de software: uma visão sobre a plataforma Java*. Elsevier Editora Ltda, 2012.

[13] Andrew Tanenbaum. *Modern operating systems*. Pearson Education, Inc., 2009.

Descrição da Interface de Programação de Aplicativos do Motor de Execução do Guaraná

Ivan E. M. Kühne*

Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brasil
oitentaetres@gmail.com

***Resumo*—O presente trabalho aborda a segunda geração do motor de execução da tecnologia Guaraná, desenvolvido como uma prova de conceito da linguagem Guaraná DSL. Ele demonstra a Interface de Programação de Aplicativos oferecida pela implementação desse motor de execução. Para tanto, são apresentados alguns trechos do código-fonte de um dos exemplos que faz parte da documentação dessa implementação, descrevendo parcialmente como ele é desenvolvido a partir das classes oferecidas por ela. Esse exemplo aborda o caso de estudo da cafeteria, que é considerado um caso clássico no campo da Integração de Soluções Empresariais.**

Palavras-chave: Integração de Soluções Empresariais; Linguagem de Domínio Específico; Prova de Conceito; Motor de Execução; Interface de Programação de Aplicativos.

I. Introdução

O campo da Integração de Soluções Empresariais (EAI) busca sincronizar dados e aplicações já existentes, além de prover novas funcionalidades a partir deles [2]. Dentro desse campo, existem diversas técnicas que foram desenvolvidas ao longo dos anos. Uma dessas técnicas é o uso de linguagens de domínio específico (DSL), que podem ser utilizadas para a modelagem conceitual das soluções de integração.

As vantagens para o uso de DSL são apontadas por Ghosh [3]. Segundo o autor, elas são semelhantes a linguagens de programação. Entretanto, cada DSL é criada para utilização em um determinado domínio de problema, estando limitada a ele. Ainda segundo ele, dentro do domínio específico de cada uma, elas são extremamente expressivas, podendo ser compreendidas até por pessoas sem familiaridade com linguagens de programação tradicionais. Além disso, a sua sintaxe e semântica modelam conceitos no nível de abstração do domínio do problema, não no nível de abstração do domínio da solução.

Conforme demonstrado pelo autor, pode ser criada uma DSL para trabalhar com o domínio das aplicações financeiras, por exemplo. Dessa forma, um analista de bolsa de valores pode descrever de forma objetiva como um software que gerencia investimentos de forma automática deve ser desenvolvido pelos engenheiros de software, sem, no entanto, precisar conhecer uma linguagem de programação como Java.

Uma das DSL desenvolvidas para o campo da EAI foi a Guaraná DSL, posteriormente transformada na tecnologia Guaraná através da criação de, entre outras coisas, um motor de execução capaz de transformar as soluções de integração modeladas conceitualmente em soluções computacionais reais.

O presente trabalho aborda a Interface de Programação de Aplicativos (API) disponibilizada pela segunda geração da versão acadêmica desse motor de execução. Para demonstrar essa API, são descritos alguns trechos adaptados do código referentes à implementação de um caso de estudo descrito por Frantz [1], disponível na documentação dos códigos-fonte.

* Bolsista PIBIC/CNPq

II. Linguagem Guaraná DSL

A linguagem Guaraná DSL foi criada com o objetivo de possibilitar a modelagem de soluções de integração em alto nível de abstração, oferecendo suporte para os padrões de integração documentados por Hohpe and Woolf [5]. Ela possibilita que os engenheiros de software modelem soluções de integração através do uso da sua sintaxe concreta, que é gráfica e extremamente intuitiva.

Associado à essa sintaxe concreta, existe um metamodelo, também chamado de sintaxe abstrata. Como componentes dessa sintaxe abstrata, temos as soluções de integração, os processos, as portas, os links, as tarefas e os slots [1].

III. Motor de Execução

A versão acadêmica do motor de execução da tecnologia Guaraná foi desenvolvida como uma prova de conceito, dentro dos esforços para permitir que as soluções de integração modeladas através da linguagem Guaraná DSL sejam transformadas em soluções computacionais reais. Esse motor de execução foi criado a partir da implementação em linguagem Java dos componentes da sintaxe abstrata da Guaraná DSL.

Essa implementação é compatível com o ambiente de desenvolvimento integrado (IDE) Eclipse. Para tanto, é preciso que a pasta com os códigos-fonte seja adicionada ao Eclipse como uma biblioteca de usuário. Após isso, quando se deseja que ela seja usada em um determinado projeto, é preciso que o projeto seja configurada de forma que a biblioteca criada esteja incluída no seu "build path".

Essa biblioteca oferece uma API composta de diversas classes, que implementam os diversos componentes da sintaxe abstrata da linguagem Guaraná DSL. A partir dessas classes, pode-se desenvolver uma solução computacional real, capaz

de integrar diferentes aplicações. Essas classes também podem ser utilizadas para a simulação de integrações, através do uso de portas mock que podem ser configuradas para substituir a comunicação com aplicações reais.

IV. Exemplo de Utilização

O exemplo descrito é baseado no problema da cafeteria descrito por Frantz [1], que se refere a como uma cafeteria pode integrar os diversos componentes do processo relativo ao processamento dos pedidos feitos pelos seus clientes. Conforme Frantz [1] apud Hohpe [4], o problema da cafeteria se tornou o padrão *de facto* para a comparação de propostas de integração do ponto de vista prático.

O exemplo encontrado na documentação mostra a criação de um Processo, componente da sintaxe abstrata da linguagem Guaraná DSL, a partir de diversos componentes disponibilizados pela API do motor de execução da tecnologia Guaraná. Desse exemplo, foram selecionados e adaptados alguns trechos que demostram como é criada e configurada a porta de entrada das mensagens no Processo, bem como a criação e configuração das suas duas primeiras tarefas.

O trecho de código a seguir demonstra a declaração e criação de dois objetos. O primeiro deles é uma pipeline responsável por receber as mensagens do sistema. O segundo objeto representa a porta de entrada das mensagens no processo, 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 de código a seguir demonstra a declaração e a criação de dois objetos, sendo que um representa uma tarefa do tipo Splitter e outro representa uma tarefa do tipo Distributor, 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);
```

Por fim, é configurada a forma como o objeto que representa a porta de entrada e os objetos que representam as instâncias das tarefas Splitter e Distributor devem trabalhar conjuntamente. As mensagens que chegam ao objeto ordersEntry são direcionadas para a entrada do objeto splitter. Em seguida, a saída do objeto splitter é direcionada para a entrada do objeto distributor, conforme o trecho de código a seguir.

```
ordersEntry.registerSlot(
  splitter.getEntrySlot(Splitter.INPUT));
splitter.bind(distributor);
```

V. Conclusão

O presente trabalho descreveu brevemente a linguagem Guaraná DSL, desenvolvida para ser utilizada no campo de EAI, e como ela foi transformada em uma tecnologia capaz de possibilitar a criação de soluções computacionais reais. Entre os componentes necessários para essa transformação, está o seu motor de execução. Foi demonstrada a API desse motor de execução, através de trechos adaptados do código-fonte de um dos exemplos que acompanham a sua documentação.

A partir disso, ficou demonstrado brevemente o processo de trabalho com as funcionalidades oferecidas pela API, que ocorre de forma simples quando já se possui a solução de integração modelada de forma conceitual. Dessa forma, basta que sejam criados os objetos referentes aos diversos componentes da solução modelada e que eles sejam configurados para trabalhar conjuntamente.

As portas que se comunicam com as aplicações externas às soluções de integração implementadas podem ser configuradas para trabalhar tanto com aplicações reais quanto com o uso de portas mock, em simulações. É possível até que as duas opções sejam utilizadas de forma simultânea. Com isso, abre-se a possibilidade para o estudo da viabilidade de determinadas soluções de integração, através do controle das características do fluxo de entrada de mensagens, especialmente quando submetidas a cenários críticos.

Referências

[1] Rafael Z. Frantz. *Enterprise Application Integration: An Easy-to-Maintain Model-Driven Engineering Approach*. PhD thesis, Universidad de Sevilla, 2012.

[2] Rafael Z. Frantz, Antonia M. Reina Quintero, and Rafael Corchuelo. A domain-specific language to design enterprise application integration solutions. *International Journal of Cooperative Information Systems*, 20(02):143–176, 2011.

[3] Debabish Ghosh. *DSLs in Action*. Manning Publications Co., 2011.

[4] Gregor Hohpe. Your coffee shop doesn't use two-phase commit [asynchronous messaging architecture]. *IEEE Software*, 22(02): 64–66, 2005.

[5] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.

Modelagem de uma Solução de Integração para Extração e Qualificação de Publicações a partir de Curriculos Lattes

Matheus H. Rehbein*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brazil
mrehbein45@gmail.com

***Resumo*—Desenvolver uma aplicação que contenha todas as necessidades de uma empresa pode se tornar custoso e então, inviável. Para suprir essa necessidade as empresas acabam utilizando um ecossistema de software heterogêneo e em algum momento da vida do software pode ser necessário adicionar novas funcionalidades ao ecossistema. A integração de aplicações surge como uma maneira de sincronizar as aplicações que não foram pensadas em ser integradas, sem modifica-las. Utilizando uma integração é possível automatizar diversos serviços que necessitam de atualizações constantes, como a atualização em websites com as publicações que estão nos currículos de pesquisadores. Este trabalho tem como objetivo realizar uma integração utilizando a ferramente de integração Apache Camel para automatizar a postagem de publicações e suas qualificações no site do GCA tendo como entrada Currículos de pesquisadores, Qualis e H-Index. Outro objetivo é criar uma modelagem conceitual do problema utilizando o Guaraná.**

Palavras-chave: Integração de Aplicações Empresariais; Apache Camel; Guaraná.

I. Introdução

Atualmente as empresas estão cada vez mais dependentes de softwares para auxiliar ou, até mesmo, realizar as tarefas empresariais do seu cotidiano. Essas tarefas podem ser realizadas por uma única aplicação, porém para que supra todas as necessidades de uma empresa, essa aplicação pode ter um desenvolvimento longo e custoso, portanto inviável [2]. Por esse motivo as empresas utilizam mais de um aplicação em seu ecossistema de software [5].

As aplicações pertencentes ao Ecossistema de Software podem ter aspectos arquiteturais diferentes, como por exemplo, linguagem de programação, base de dados, plataforma, etc. Em algum determinado momento uma aplicação pode vir a necessitar de uma comunicação ou processo que está contida em outra, essa interação é conhecida como integração de aplicações empresariais (EAI).

Para realizar a integração existem tecnologias, elas auxiliam o desenvolvedor. Esse processo deve ser bastante abstrato, ou seja, se aplicação "A" for totalmente distinta da aplicação "B", a integração deverá funcionar da mesma maneira. Entre as tecnologias destacam-se: Apache Camel, Guaraná, Mule e Spring Integration [2]. Essas quatro ferramentas utilizam o padrão de integração por mensagens citado por Hohpe e Woolf [3].

Muitos serviços repetitivos podem ser automatizados com a integração entre aplicações. Um exemplo é a fazer com que os dados de um determinado pesquisador seja extraído a partir de seu currículo lattes, combinado com outras informações (qualificações dos trabalhos), e então publicado em portais web. Este trabalho tem como objetivo apresentar uma integração entre o currículos lattes de pesquisadores, qualificações de trabalhos e portais web utilizando a ferramenta Apache Camel, além de modelar o problema de integração com a tecnologia Guaraná.

II. Caso de Estudo

A. Ecossistema de Software

A intenção deste trabalho é ter como entrada currículos lattes de pesquisadores obtidos a partir do site do CNPQ no formato de XML. Em seguida será extraído as informações do Qualis e H-Index das publicações, essas informações serão adicionadas ao currículos. Por fim, será gerado um arquivo HTML com todas as publicações e qualificações, que o usuário desejar.

B. Modelo Conceitual

Para que não aconteça erros, é possível criar uma descrição do problema, essa descrição é conhecida como modelagem conceitual. Essa modelagem apresenta os componentes que serão utilizados durante a integração. Neste trabalho apresenta o modelo desenvolvido para para a solução de integração proposta.

A Figura 1 apresenta o modelo conceitual da integração. É importante ressaltar que cada currículo irá ser uma mensagem, portanto, se a aplicação tiver como entrada 5 currículos terá 5 mensagens. A tarefa 1 é conhecida como "Filter", ele é responsável pela filtragem dos currículos, ou seja, se tiver um arquivo que não seja um currículo lattes na pasta, esse arquivo não será utilizado. Já a 2, denominado "Replicator" tem como objetivo encaminhar uma cópia de cada currículo para outros diretórios. Em seguida é utilizado as tarefas 3, 8 e 9, chamados de "Translator", como o próprio nome já

* Bolsista PROBIC/Fapergs

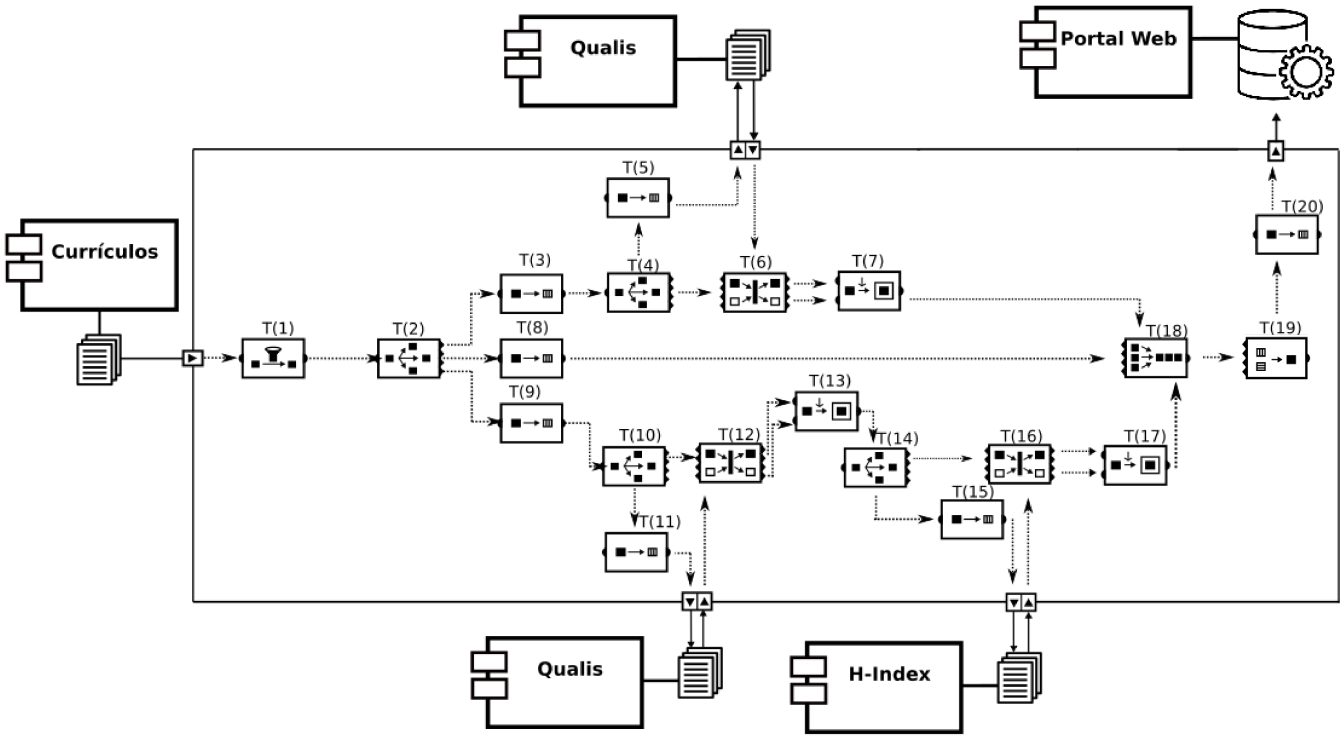

Figura 1. Modelo Conceitual da Solução deIntegração

diz, eles são utilizado para traduzir um determinado conteúdo, por se tratar de arquivos XML utilizou-se a linguagem XSLT para realizar a tradução. A 3 será responsável pelos Artigos publicados em Eventos, a 8 pelos Livros e Capítulos de Livros publicados e a 9 pelos Artigos publicados em periódicos. Em seguida está a tarefa 4, novamente um Replicator, ela tem como objetivo enviar uma cópia para a 5 que irá traduzir para uma mensagem que o Qualis irá identificar, e também para a 6 que é conhecido como "Correlator"e possui o objetivo de correlacionar mensagens. A 6 irá correlacionar a mensagem enviada pelo Qualis e a mensagem replicada pela 4, e por fim a 7, conhecida como "Context-based Content Enricher", ela irá adicionar os conteúdos obtidos na 6 ao corpo da mensagem. Posteriormente está a tarefa 18, conhecida como "Merger". Essa tarefa irá concatenar as mensagens obtidas das tarefas 7, 8 e 17 em uma única mensagem. Em seguida, terá a 19, chamada de "Assembler", ele irá construir uma nova mensagem a partir das mensagens obtidas na 18. Por fim estará a tarefa 20, que tem o objetivo de fazer com que seja gerado um HTML que o site irá reconhecer.

O processo executado entre as tarefas 3 e 7 foi repetido novamente entre as tarefas 9 e 17, porém irá obter o Qualis e H-Index do artigos publicados em periódicos.

C. Implementação

Após realizar o modelo conceitual, foi feita a implementação da solução utilizando a Fluent API [1] que o Camel disponibiliza [4]. Além do conhecimento da tecnologia de integração, também é necessário conhecer a linguagem XSLT, que é utilizada nos Translators.

III. Conclusão

Com o Guaraná tornou-se viável a modelagem conceitual da solução. Após a modelagem foi possível implementá-la com o Camel. O problema de integração é real e está presente no cotidiano de diversos pesquisadores. Para automatizar este processo somente precisará dar como entrada o seu Currículo lattes e então terá como saída uma página HTML que poderá ser editada para ficar compatível com o site que será postado. Se posteriormente for desejável adicionar novas funcionalidades a aplicação, será totalmente possível, devido a baixa manutenção que a integração proporciona.

Referências

[1] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[2] Rafael Z Frantz and Rafael Corchuelo. A software development kit to implement integration solutions. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1647–1652. ACM, 2012.

[3] Gregor Hohpe and Bobby Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[4] Claus Ibsen and Jonathan Anstey. *Camel in action*. Manning Publications Co., 2010.

[5] David G Messerschmitt, Clemens Szyperski, et al. Software ecosystem: understanding an indispensable technology and industry. *MIT Press Books*, 1, 2005.

Proposta de um Modelo de Simulação usando Teoria das Filas

Félix Hoffmann Sebastiany*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Santa Rosa, RS, Brasil
felixsebastiany@hotmail.com

***Resumo*—Este trabalho descreve o desenvolvimento de um modelo de simulação de uma solução de integração utilizando Teoria das Filas. O modelo de integração foi desenvolvido pela tecnologia Guaraná. O modelo de simulação foi desenvolvido através da ferramenta *MatLab/Simulink* com o *toolkit* denominado *SimEvents*. Com os resultados iniciais do modelo de simulação, foi possível analisar o comportamento da solução de integração.**

Palavras-chave: Integração de Aplicações Empresariais, Simulação, *SimEvents*, Teoria das Filas.

I. Introdução

Geralmente as empresas possuem várias aplicações com diferentes funcionalidades, formando um ecossistema de *softwares*. Estas aplicações geralmente são heterogêneas, e foram desenvolvidas sem levar em conta sua integração, gerando assim redundância de dados [7]. Como as mudanças nos processos de negócio das empresas são constantes, a comunicação entre diferentes aplicações tornar-se uma necessidade. Neste contexto, surge a área da computação chamada Integração de Aplicações Empresariais (EAI - *Enterprise Application Integration*), tendo como objetivo a sincronia das aplicações de um ecossistema de *softwares*. Normalmente, as etapas de desenvolvimento de software são constituídas pelas etapas de especificação, projeto, implementação, validação e evolução [6]. Entretanto, os erros de uma aplicação são detectados após a sua implementação, implantação e testes. Tais etapas são custosas, o que onera o valor final da aplicação. Como uma integração de aplicação é, também, uma aplicação que passa pelas mesmas etapas de desenvolvimento, busca-se, neste trabalho desenvolver um modelo de simulação usando Teoria das Filas a partir de um modelo conceitual, visando detectar gargalos de performance ainda na fase de projeto. Esse modelo é desenvolvido e simulado utilizando a ferramenta MatLab/Simulink/SimEvents.

O artigo está organizado da seguinte maneira: na seção II apresenta-se o referencial teórico com os principais conceitos utilizados. A seção III traz o caso de estudo. Na seção IV são apresentadas as considerações finais e trabalhos futuros.

* Bolsista PIBIC/CNPq

II. Referencial Teórico

A. Tecnologia Guaraná DSL

É uma linguagem de domínio específico da tecnologia Guaraná que proporciona ao engenheiro de *software* desenvolver soluções de integração em alto nível de abstração, utilizando uma sintaxe concreta gráfica e conceitos documentados por Gregor Hohpe e Bobby Woolf [7]. Os modelos desenvolvidos com esta linguagem são independentes das tecnologias de integração voltadas à implementação e podem ser transformados automaticamente a código de uma ou outra tecnologia. Tal característica permite que engenheiros de *software* centrem seus esforços na criação de modelos para a solução do problema, reduzindo os custos envolvidos no processo de aprendizagem e uso das distintas, e muitas vezes complexas, tecnologias voltadas à implementação [1].

B. Simulação

De acordo com Kelm [7], a simulação é um método que utiliza um modelo matemático para possibilitar o estudo e a análise do comportamento de um sistema sem que seja necessário realizar alterações no sistema real. Pode-se assim, prever o comportamento e ajudar no processo de tomada de decisão. Hillier e Lieberman [3] afirmam que a simulação não deve ser utilizada quando existe a possibilidade de um procedimento menos oneroso, capaz de fornecer as mesmas ou melhores informações sobre o sistema. Geralmente a simulação apenas é utilizada quando o sistema estudado é muito complexo para ser analisado por modelos matemáticos, de forma satisfatória.

C. Ferramenta SimEvents

O *SimEvents* é um *software* que auxilia na simulação de eventos discretos, incorporado no *Simulink* da plataforma *MatLab*, que fornece uma biblioteca de componentes para analisar modelos de sistemas e otimizá-los. Ele possui uma interface gráfica na qual o usuário pode modelar simulações a partir de blocos gráficos predefinidos, que sozinhos ou em conjunto representam tarefas de um modelo conceitual de solução de integração.

D. Teoria das Filas

De acordo com Preissler [4], um sistema de filas pode ser representado por diferentes modelos, tendo elementos característicos comuns a todos que fazem parte do processo básico,

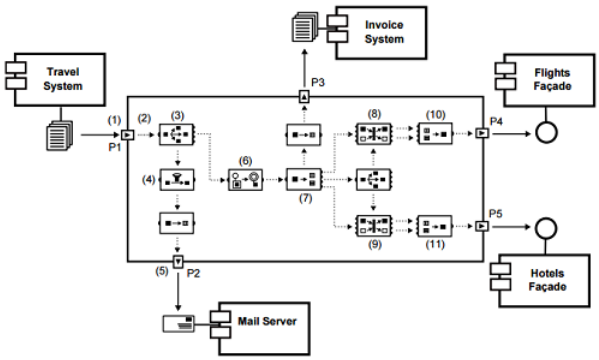

Figura 1. Modelo Conceitual de Integração *Travel System*[2].

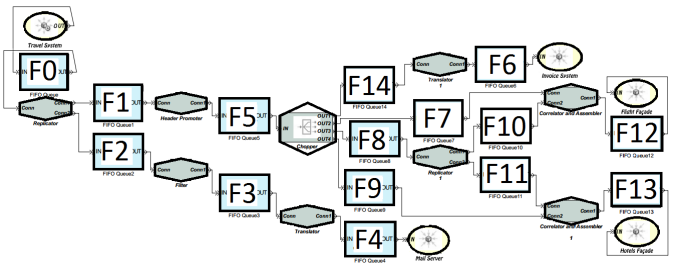

Figura 2. Modelo de Simulação desenvolvido na ferramenta MatLab/Simulink/SimEvents equivalente ao Modelo de Integração *Travel System*.

ela permite calcular e estimar resultados relacionados à performance dos sistemas, com base em propriedades mensuráveis. Uma Solução de Integração é constituída por várias filas de troca de mensagens e, segundo Kelm [7], quando as filas de um sistema recebem valores além dos adequados, passam a constituir gargalos de desempenho. O ideal seria dimensionar os sistemas de modo que não existissem ou tenham o mínimo possível de filas. Para isso surge a Teoria das Filas, que é um ramo da probabilidade que estuda a formação de Filas por meio de fórmulas matemáticas diminuindo as implicações de modo que torne o sistema de filas mais eficiente.

III. Caso de Estudo

O modelo de simulação proposto é baseado no cenário *Travel System*, trazido pela Figura 1, introduzido por Frantz [2], composto por um ecossistema de cinco aplicações integradas. Esse modelo conceitual de integração descreve um sistema de viagens, em que o cliente pode consultar os voos e hotéis disponíveis em um determinado dia (*softwares Flights Façade* e *Hotels Façade* respectivamente). Além disso, o cliente paga suas contas usando seus cartões de crédito (*software Invoice System*) e é notificado com informações sobre suas reservas por *email* através do *software Mail Server*.

Para tornar possível a simulação da solução de integração, foi realizado um estudo para obter equivalências entre tarefas do modelo conceitual de integração e tarefas da ferramenta de simulação *SimEvents*. A partir disso foi realizado o modelo de simulação conforme a Figura 2, baseado no cenário *Travel System*. Nesse modelo, podemos observar blocos de formatos distintos. Cada um dos blocos representa uma tarefa/função diferente no modelo de integração. Os círculos representam as aplicações do sistema, os retângulos representam as filas do sistema e por fim, os hexágonos representam as tarefas do sistema. Para a validação do modelo de simulação formal proposto, utilizou-se a técnica de intuição de especialistas.

IV. Considerações Finais e Trabalhos Futuros

Com a solução de integração gerada na tecnologia Guaraná DSL, foi possível elaborar um modelo formal de simulação equivalente através da ferramenta *MatLab/Simulink/SimEvents* que foi validado por intuição de especialistas. Conforme Sawicki [5] uma solução de integração pode ser caracterizada como um sistema cujo modelo é classificado como estocástico, dinâmico e discreto. Com a modelagem de simulação foi possível analisar o comportamento da solução de integração, prevendo quais caminhos serão tomados.

Em trabalhos futuros, será caracterizado cenários de simulação para uma exploração mais detalhada nos resultados da simulação, identificando possíveis gargalos de desempenho ainda na fase de projeto e também, a aplicação de fórmulas matemáticas da Teoria das Filas para possivelmente otimizar as soluções de integração de aplicações empresariais.

Referências

[1] Rafael Frantz, Sandro Sawicki, Fabricia Roos-Frantz, Rafael Corchuelo, Vitor Basto-Fernandes, and Inma Hernández. Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem. *XIX Jornada de Pesquisa*, pages 1–11, 2014.

[2] Rafael Zancan Frantz. Enterprise application integration: An easy-to-maintain model-driven engineering approach. 2012.

[3] Frederick S Hillier and Gerald J Lieberman. *Introdução à pesquisa operacional*. McGraw Hill Brasil, 2013.

[4] Amanda Preissler and Sandro Sawicki. Modelo de simulação de uma solução de integração teórica utilizando a ferramenta matlab/simulink. *IV SFCT*, 7:28, 2016.

[5] Sandro Sawicki, Rafael Z Frantz, Vitor Manuel Basto Fernandes, Fabricia Roos-Frantz, Iryna Yevseyeva, and Rafael Corchuelo. Characterising enterprise application integration solutions as discrete-event systems. In *Handbook of Research on Computational Simulation and Modeling in Engineering*, pages 261–288. IGI Global, 2016.

[6] Ian Sommerville. *Software Engineering (5th Ed.)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.

[7] Arléte Kelm Wiesner. Modelagem matemática e computacional para análise do comportamento de soluções de integração de aplicações através da criação de modelos de simulação com a teoria das filas. *UNIJUI - Universidade Regional do Noroeste do Estado do Rio Grande do Sul-Ijuí/RS*, 2015.

A Comparison of Petri Net Simulation Tools

João Pedro L. Petter
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brazil
jplpetter@gmail.com

***Resumo*—Nowadays it has become almost a norm for enterprises to have a well-designed software ecosystem, in order to save resources and even make some tasks automatic. It is a big risk for companies to just put a program into a software ecosystem, for it can disrupt other applications and stagnate employees, which is why computer simulation exists. Utilizing the ingenious concept of Petri nets simulation tools are extremely useful. However, there are many simulation tools out there. It is in a company's best interest to select a tool that can cater to its needs, for software ecosystems are incredibly divergent. In this article, we will look at three Petri net simulation tools, Platform Independent Petri net Editor 2 (PIPE2), CPN tools and HPsim, and compare them using a comparison framework devised by Kraisig, Welter and Frantz. The aforementioned framework is simple and was devised from the perspective of application integration.**

Palavras-chave: Petri nets; Simulation.

I. Introduction

In today's ever more competitive and technological business world, it is important, perhaps more than ever, for companies to secure functional programs and applications in order to assist its employees, thusly saving important time and resources. Frequently new applications are introduced to a company's software ecosystem, which can be catastrophic should the aforementioned application fail, possibly disrupting the work of many employees, consequently consuming the company's resources and even damaging other applications in the software ecosystem. Because of this computer simulation has become a common process for enterprises large and small. Computer simulation allows for safer software integration, therefore protecting the companies resources, making for a cheaper alternative to experimenting with the real application and with no real risks but at the expense of extra time consumed. Many tools use Petri nets as a base for simulation as a result of the ease in which Petri nets can model the characteristics of a system and how well its model lends itself to discrete event simulation. Petri nets were created in 1962 by Carl Adam Petri and were initially used as a way to describe a chemical reaction. Nowadays Petri nets are used mainly for computer simulation. "Petri nets (Petri 1962, Peterson 1981) are a wellfounded process modeling technique that have formal semantics. They have been used to model and analyze several types of processes including protocols, manufacturing systems, and business processes(Aalst 1999). A Petri net is a directed, connected, and bipartite graph in which each node is either a place or a transition. Tokens occupy places. When there is at least one token in every place connected to a transition, we say that the transition is enabled. Any enabled transition may fire removing one token from every input place, and depositing one token in each output place." [1]. In this paper, we are going to compare three Petri net simulation tools using a comparison framework created by Kraisig, Welter and Frantz (2016) [3]. We will compare the Platform Independent Petri net Editor 2 (PIPE2), CPN tools and HPsim; we will revise the elements of each comparison before presenting a table and a conclusion, where we will discuss the results of the comparison. Note that the function comparison is absent in this article.

II. Tools

In this paper, we are going to compare three Petri net simulation tools using a comparison framework created by Kraisig, Welter and Frantz (2016) [3]. We will compare the Platform Independent Petri net Editor 2 (PIPE2), CPN tools and HPsim; we will revise the elements of each comparison before presenting a table and a conclusion, where we will discuss the results of the comparison. Note that the function comparison is absent in this article. The criteria of the metrics used in this comparison framework are based on how easily the programs can be integrated in a software ecosystem, as well as the general flexibility of the tool. This metrics are: The type of petri net supported, which determinates the usefulness of the tool in certain situations, the components supported by the tools, in which ambient can the tools operate, this is very important, for some tools won't run on some operating systems, how accessible is the user interface, in other words, how much depth there is to the tool and how long will it take for someone to fully understand its functionalities, how complex are the results of the simulation, and what are the features from the tools editor.

III. Results

All the three tools we are about to measure use Petri nets in order to perform simulations, so as there are several types of Petri nets it is imperative for us to contrast which tools can use what type of Petri net. First, there is the basic Petri net in all of its simplicity, all the three tools can operate with the basic Petri net. Stochastic Petri nets are nets with random delays between transitions, these are supported by PIPE 2 and HPsim. Colored Petri nets allow a data value to be attached to a token, CPN Tools specialize in colored nets, even been named after them(Colored Petri Net Tools), but PIPE 2 supports

them as well. “Coloured Petri nets and Predicate/transition-nets are very closely related to each other, in the sense that Coloured Petri nets have been developed as a modification of Predicate/transition-nets, in order to avoid some technical problems which arise when the method of place-invariants is generalized to apply for Predicate/transition-nets.” [2]. Timed Petri nets are nets that incorporate the concept of time, firing transitions in accordance to a timer, this is essential for uniform environment modeling. All three tools support timed nets. Hierarchical Petri nets allow for the creation of large models using a small number of nets, for they allow Petri nets to be placed inside tokens, working as a net inside a net. PIPE 2 and CPN Tools support hierarchical nets. The components are an important factor in the functionality of the tools, only CPN Tools allows user code to modify the tools functionality. All three tools have a graphical editor, token game animation, and all three allow for simple performance analysis, although CPN Tools boasts a series of monitoring options that allows for more complex results while PIPE 2 has structural, archive format and extensive modulus analysis. PIPE 2 permits graphical simulation and structural analysis while CPN Tools and HPsim bear fast simulation. Both PIPE 2 and CPN Tools allow for state spaces and interchange file format. PIPE 2 operates on java and thus can run on most machines that support java. CPN Tools and HPsim both work on the Windows operating system, while CPN Tools has a flawed and substantially inferior and Linux port but can run on a Linux or a MAC as long as Windows is emulated. It is very important, especially for new users, that a work interface and results are simple and easy to understand. HPsim’s interface is remarkably simple and one could learn how to use the tool in less than an hour. PIPE 2 is also very simple if slightly more convoluted HPsim’s, it’s possible to grasp the tool in very little time. CPN Tools interface is labyrinthian and overly complex, it can take an entire day to learn the bare minimum to operate said tool and much longer to master how to correctly insert user code in it. PIPE 2 and HPsim provide simple results, so does CPN Tool, should one not use the plethora of optional monitors in a simulation. Finally, there are the editing tools of these simulators. All three provide zoom and editing. PIPE2 and CPN Tools support undo and redo functions. CPN Tools allows for the cloning of and entire net while PIPE 2 and HPsim are stuck with copy and paste. HPsim boasts text annotations and a print function. PIPE 2 has auto adhesive notes. CPN Tools allows for animations.

[1]

IV. Conclusion

Based on the data gathered and studied during this article, it is possible to observe that PIPE 2 is the most practical of the trio due to its numerous components and the large variety of Petri net types it supports. However, HPsim is simpler and faster, making It a better option for less complex tasks and for introducing beginners to the concept of Petri nets. While nightmarish complex CPN Tools support of user code and focus on colored Petri nets guarantee that the tool has its uses, not to mention the monitors that can be employed in a simulation, making it a decent option for in-depth analysis. CPN Tools focus in colored nets works for its benefit, it makes the aforementioned tool a more specialized option that can be expanded with user code, even tough PIPE 2 is more pragmatic and interpretative. Ultimately we can see that all three Petri net simulation tools viewed in this article have their uses in a specific situation.

[1]

** Special thanks to Rafael Z. Frantz and Sandro Sawicki for orientation

	Pipe 2	CPN Tools	HP Sim
Petri Net Types	Basic, stochastic, colored, timed and hierarchical	Basic, colored, timed, hierarchical	Basic, stochastic, timed
Components	Graphical editor, token game, graphical simulation, state spaces, Invariant place, structural analysis, simple performance analysis, interchange file format, extensive modulus analyses, archive format analyses	Allows user code, graphical editor, Fast Simulation, token game animation, state spaces, simple performance analysis, interchange file format	Graphical editor, token game animation, fast simulation, simple performance analyses
Environments	Java	Linux(inferior), Windows	Windows
Work Interface	Assessable	Complex	Assessable
Simulation Results	Simple Results	Simple and Complex results	Simple results
Editor	Zoom, exportation, editing, Auto-adhesive notes, undo, redo	Zoom, editing, animation, undo, redo, cloning	Zoom, print function, text annotations, editing

Figura 1. table 1.

Referências

[1] Rachid Hamadi and Boualem Benatallah. A petri net-based model for web service composition, 2003.
[2] K. Jensen. Coloured petri nets, 1987.
[3] Adriana Rosélia Kraisig, Franciéli C. Welter , and Rafael Z. Frantz. Framework de comparação entre ferramentas de simulação. 2016.
[4] M. Ajmone Marsan. Stochastic petri nets: An elementary introduction. 2007.
[5] Will van der Aalst. *Simulation-based performance and reliability analysis of business processes*. 2013.

Comparação do Esforço Empregado para o Aprendizado de Plataformas de Integração de Aplicações a partir de um Estudo Empirico

Thainan H. Feistel*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brazil
qthainan@gmail.com

***Resumo*—As empresas estão cada vez mais crescendo em tamanho, com isso o número de aplicações necessárias para gerenciar a mesma também cresce. Como essas aplicações são em grande parte adquiridas de empresas terceirizadas, as implementações são distintas, fazendo assim com que elas não consigam trocar informações sem a realização de mudanças. A partir disso se faz necessário a utilização de ferramentas de integração, com o intuito de construir soluções de integração capazes de possibilitar a troca de dados e funcionalidades entre as distintas aplicações da empresa. Estas ferramentas possuem um tempo de aprendizagem distinto, e este estudo comparativo tem como objetivo dar uma ideia do tempo de aprendizagem de três ferramentas específicas, as quais foram estudadas e avaliadas por três desenvolvedores, cada um utilizando uma ferramenta apenas.**

Palavras-chave: Integração de Aplicações Empresariais; Ferramentas de Integração; Estudo Comparativo.

I. Introdução

A maior parte das empresas atualmente dependem de suas aplicações internas para realizar grande parte de suas atividades. Estas aplicações normalmente são compradas de empresas terceirizadas, o que resulta em uma coleção de aplicações distintas, tornando impossível a realização de troca de dados e funcionalidades entre as aplicações. A área de integração de aplicações empresariais tem como objetivo principal fornecer metodologias e ferramentas para que aplicações heterogêneas consigam trocar dados e funcionalidades sem que seja necessário realizar mudanças nas aplicações em si. Esta troca de dados e funcionalidades é realizada através de uma solução de integração que atua entre as duas ou mais aplicações de uma empresa, sendo que se essa solução for desconectada, todas as aplicações continuam trabalhando de forma individual normalmente, devido ao fato da solução de integração não modificar as aplicações em si para que haja a troca de informações.

As soluções de integração são implementadas a partir da utilização de ferramentas de integração. Existem diversas ferramentas de integração no mercado atualmente, sendo que o nível de aprendizagem delas varia bastante, pois algumas são mais simples e abordam menos aspectos em geral de integração. Já outras são mais complexas pois abordam praticamente todos os aspectos de integração e também pelo fato de focar em uma área específica do mercado de trabalho. Estas ferramentas mais complexas demandam um tempo de estudo maior devido ao seu tamanho e nível de complexidade.

Algumas das ferramentas de integração são: Apache Camel [3], Mule [1] e Spring Integration [4]. Estas três ferramentas seguem os mesmos padrões de integração que foram fornecidos por Gregor Hohpe e Bobby Woolf no livro Enterprise Integration Patterns [2]. O código das ferramentas é open-source e elas tem uma área de abrangência geral, não se especializando em áreas específicas do mercado de trabalho.

Estas três ferramentas foram selecionadas para um estudo comparativo entre os seus respectivos tempos de aprendizagem. Algumas métricas foram definidas para a coleta dos tempos, como a instalação da ferramenta em si, e a implementação de alguns exemplos básicos. O estudo prático e a coleta dos tempos foi realizada por três desenvolvedores, sendo que cada um trabalhou com uma das ferramenta apenas. Os três desenvolvedores possuem experiência em Java, conhecimento da área de integração de aplicações empresariais, mas não possuíam conhecimento das ferramentas em específico.

II. Ferramentas de Integração

A maioria das ferramentas de integração de aplicações seguem os padrões, metodologias e modelos descritos por Gregor Hohpe e Bobby Woolf no livro Enterprise Integration Patterns [2]. Nele são descritos diferentes estilos de integração, diversos conceitos de adaptadores e canais que são utilizados para receber e entregar dados com formatos distintos e também compartilhar funcionalidades entre as aplicações.

As três ferramentas utilizadas no estudo comparativo são open-source e seguem os mesmos padrões de integração, apenas variando o nome dos componentes utilizados na implementação da solução. É possível utilizar as três ferramentas dentro de um ambiente de programação Java, tendo apenas que importar os pacotes necessários para cada ferramenta. No entanto, o Spring Integration e o Mule possuem uma aplicação própria para baixar e utilizar, a do Spring Integration é uma extensão de um ambiente Java com os pacotes já importados,

* Bolsista PIBITI/CNPq

MÉTRICAS	Apache Camel	Mule ESB	Spring Integration
Tempo para configuração do ambiente de desenvolvimento	04:30	01:30	05:00
Tempo para elaborar um exemplo básico na tecnologia	00:50	00:40	00:30
Tempo para implementar exemplo de solução simples (que move arquivo XML de um diretório para outro)	00:20	00:30	00:10
Tempo para implementar exemplo de solução simples (que copia arquivo XML de um diretório para outro)	01:00	03:00	02:00
Tempo para estudar e configurar o adaptador XML de *leitura* para o exemplo	03:00	01:00	02:00
Tempo para estudar e configurar o adaptador XML de *saída* para o exemplo	01:00	01:00	02:00
Tempo para estudar e aplicar Splitter em XML	04:00	03:30	03:00

Figura 1. Tabela das Métricas do Estudo Comparativo.

e também são fornecidos alguns exemplos juntos com tutoriais para uma aprendizagem básica mais rápida. A aplicação disponível do Mule é mais avançada, nela são fornecidos todos os pacotes necessários e também uma interface de drag and drop, o que facilita bastante na aprendizagem, e também proporciona um aumento na velocidade da implementação.

III. Metodologia

Cada desenvolvedor realizou um estudo inicial da sua respectiva ferramenta, com o intuito de coletar informações para auxiliar no estabelecimento do ambiente e ter uma maior facilidade na hora de começar a implementar com a mesma.

Diversos artigos e livros da área de integração e das ferramentas em específico foram estudados para aprender mais sobre os padrões de integração e também como eles seriam retratados nas ferramentas, pois cada uma possui uma nomenclatura diferente para suas funcionalidades.

Na parte da implementação dos exemplos contidos nas métricas, foi realizado um estudo de quais funcionalidades se precisaria usar, e também de como usar a mesma. Após isso era realizada a implementação do exemplo. Fóruns disponibilizados pelas ferramentas foram de grande ajuda nessa etapa, pois a maioria das dúvidas que apareceram já haviam sido respondidas antes. Também foram utilizados livros que continham tutoriais de soluções simples, onde era possível ver como a ferramenta e suas funcionalidades funcionavam, conseguindo assim adaptá-las para utiliza-las nos exemplos das métricas em estudo.

IV. Análise Comparativa

A figura 1 mostra todas as métricas que foram determinadas e os tempos registrados a partir do estudo e implementação dos exemplos propostos. Cada uma das métricas leva em conta o tempo de estudo e o tempo de implementação que foi necessário.

O tempo de configuração do ambiente de desenvolvimento foi praticamente igual entre Apache Camel e Spring Integration, pelo fato de as duas ferramentas poderem utilizar um ambiente de programação Java para implementar, já Mule tem uma aplicação própria e de fácil acesso em sua página web, o que deixa bastante simples o processo de configuração inicial, tendo assim um tempo menor de instalação.

O exemplo básico utilizado foi o mesmo em todas as tecnologias e pode-se notar que não houve uma grande diferença de tempo nessa métrica.

Já na métrica de copiar um arquivo de um diretório para outro, Spring Integration e Mule tiveram um tempo maior, sendo que Spring Integration teve o dobro de tempo do Apache Camel e o Mule três vezes o tempo do Apache Camel. A métrica de mover de um diretório para outro é bastante similar a de copiar um arquivo, só foi necessário alterar para excluir o arquivo fonte, o que não necessitou de um estudo específico, deixando o tempo bastante baixo e igual entre as ferramentas.

Nas métricas de leitura e escrita de um arquivo XML, Mule se saiu melhor, pelo fato de utilizar o mesmo conector para fazer ambos serviços, Apache Camel e Spring Integration ficaram com tempos maiores por utilizarem conectores diferentes, necessitando de um tempo maior de estudo.

Por fim, a métrica de configurar um adaptador Splitter em XML, que consiste em separar um arquivo XML utilizando uma expressão XPath, com o objetivo de criar um novo arquivo XML com apenas a parte desejada do antigo. Os tempos dessa métrica foram bastante similares, sendo que Spring Integration obteve o menor tempo, porém as outras ferramentas ficaram bastante próximas, com apenas uma hora de diferença no tempo mais distante.

V. Conclusão

As empresas atualmente estão cada vez mais obtendo novas aplicações para realizar seu trabalho interno, estas aplicações são heterogêneas entre si, o que impossibilita a troca de dados e funcionalidades entre as mesmas. Essa troca é de extrema importância dentro da empresa, pois cada aplicação atende a um setor em específico, e a troca de informações entre os setores é necessária. A partir disso é necessário utilizar ferramentas de integração para implementar soluções que possibilitem essa troca de informações, para isso é necessário ter o conhecimento do tempo de aprendizagem das ferramentas para iniciar a implementação o quanto antes. Com este estudo comparativo é possível se ter uma ideia de qual ferramenta necessita de um tempo menor de aprendizagem, facilitando na escolha da mesma quando necessário.

Referências

[1] David Dossot, John D'Emic, and Victor Romero. *Mule in action*. Manning, 2014.

[2] G. Hohpe and B.A. WOOLF. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. The Addison-Wesley Signature Series. Prentice Hall, 2004.

[3] Claus Ibsen and Jonathan Anstey. *Camel in action*. Manning Publications Co., 2010.

[4] Craig Walls. *Spring in Action*. Manning, Shelter Island, 2011.

Análise de Métricas de Manutenibilidade de um Sistema de Software de Integração: Mulesoft

Rodolfo Berlezi*
Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Ciências Exatas e Engenharias
Ijuí, RS, Brazil
rodolfo_berlezi@hotmail.com

***Resumo*—Com o passar dos anos as organizações empresariais vem necessitando cada vez mais de integrações de sistemas, sendo necessário a utilização de uma ferramenta adequada. Estas ferramentas geralmente estão em constante evolução de suas versões para serem otimizadas ou ampliar suas possibilidades de integração. Existe a possibilidade, através de métricas pré estabelecidas, de averiguar o quão tem sido melhorado o software a cada nova versão, assim realizamos um estudo de análise das evoluções de versão do Mulesoft com base nestas métricas.**

Palavras-chave: Mulesoft, Integração de Aplicações Empresariais, Manutenibilidade de Software.

I. Introdução

Na atualidade o ambiente empresarial possui diversos setores podendo cada setor ter um diferente software que é responsável pela realização e controle de suas respectivos tarefas. Assim com o passar dos anos uma diversidade de softwares, tanto os feitos pela própria empresa ou por empresas terceirizadas, criam o chamado ecossistema da empresa. Cada software vem com diferentes necessidades e compatibilidades por isso necessitam de um meio de comunicação e integração de suas informações, que facilmente são incompatíveis entre eles visto que foram desenvolvidos sem pensar em tal necessidade. Então surgem os softwares especificamente feitos para realizar esta função, são eles, os sistemas de softwares de integração, tais como Apache Camel, Spring Integration, Mulesoft e Guaraná DSL.

Segundo a IEEE [3], existem três tipos diferente de manutenção: Corretiva, Perfectiva e Adaptativa. A manutenção adaptativa é aquela executada em ordem para remodelar o sistema, adicionando novas possibilidades do mesmo ser executado em diferentes ambientes ou submeter processos que não foram inicialmente configurados para o sistema, sendo esta manutenção a escolhida para avaliação no artigo. Já a ferramenta escolhida para está pesquisa é o Mulesoft, sendo uma das ferramentas citadas no catálogo de integração de Hohpe e Woolf [2] de código aberto e concebendo todos os conceitos de integração. Utilizamos distintas versões para uma analise comparativa da evolução do software.

O artigo está organizado em três grandes partes. Primeiramente a metodologia onde será explicado como foi feita a coleta dos dados. Depois um embasamento teórico sobre o significado de cada dado e após uma análise profunda dos dados e resultados obtidos. Por fim uma conclusão.

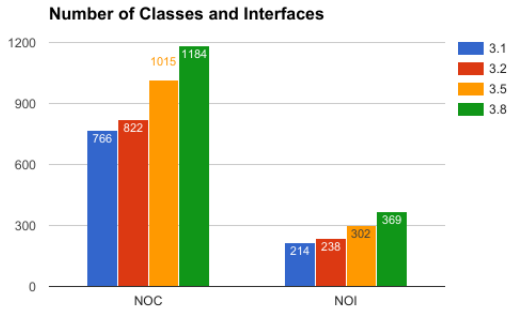

Figura 1. Gráfico da Métrica de Classes e Interfaces

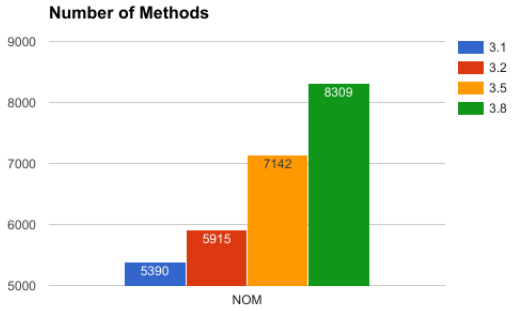

Figura 2. Gráfico da Métrica de Métodos

II. Metodologia

Para averiguar a evolução dos graus de manutenção no Mulesoft nos limitamos escolhendo quatro versões distintas em ordem evolutiva do Mule: 3.1, 3.2, 3.5 e 3.8.

As métricas usadas para medir o grau foram tiradas de pesquisas anteriores baseadas em Lanza e Marinescu [4], onde suas métricas foram classificadas em quatro grandes grupos: Medidas de tamanho, de acoplamento, de complexidade e de

* Bolsista PIBIC/UNIJUÍ

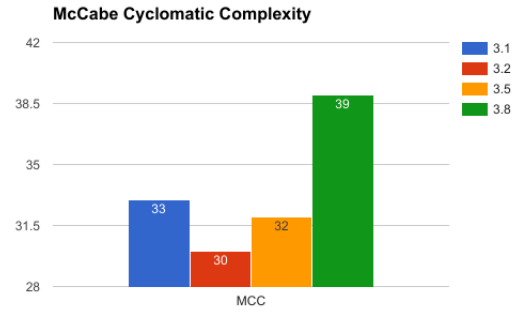

Figura 3. Gráfico da Métrica da Complexidade Ciclomática de McCabe

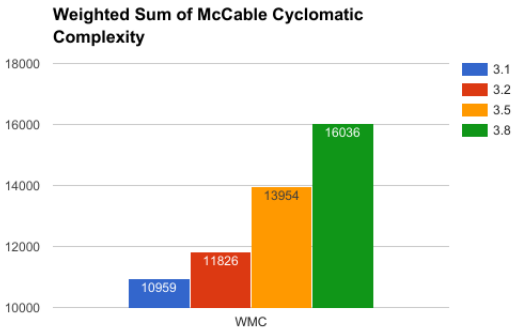

Figura 4. Gráfico da Métrica da Soma dos Pesos da Complexidade Ciclomática de McCabe

herança. As mesmas puderam ser obtidas através do uso do software Metrics, nos entregando as informações e dados do código fonte de cada versão em tabelas. Neste artigo demos o foco para as medidas de tamanho e de complexidade que serão explicadas a seguir.

III. Embasamento Teórico

Medidas de tamanho são aquelas de quantidade, sobre quantos pacotes, classes, interfaces, métodos, parâmetros e linhas tem o código do software. Estas medidas nos mostram o quão grande o sistema é. Em especial, destacaremos a quantidade de classes, interfaces e métodos do sistema.

A quantidade de classes e interfaces podem indicar o quanto de esforço é necessário para entender como o sistema de pacotes é organizado e ainda promove uma visão geral de todo o design do sistema. Já a quantidade de métodos tanto em classes como em interfaces, é uma métrica que pode ser usada para indicar o potencial reuso de uma classe. Como Frantz [1] mostra, uma vasta gama de métodos demostram o quanto uma classe está para uma aplicação especifica, o que limita o seu reuso.

Focando na manutenção do software a complexidade dele diz muito. As próximas medidas indicam o quão complexo e difícil pode ser entender e realizar manutenções no código. O principal meio de medir é pela complexidade ciclomática de McCabe [5] e a soma de seus pesos.

A soma dos pesos da complexidade ciclomática de McCabe é para todos os métodos em uma classe indicando o quão difícil será entender e depois modificar os mesmos. O quão maior for este valor, mais esforço é necessário para a aplicação de manutenção na classe. Logo a complexidade ciclomática de McCabe [5] indica o quão complexo é o algoritmo em um método, sendo recomendado que a medida não ultrapasse o valor de dez, já que está diretamente relacionado com a dificuldade de manutenibilidade de um pedaço do software, sendo um fator decisivo para a mesma.

IV. Análise de Resultados

Como é averiguado nas Figuras 1 e 2, podemos ver que o sistema de software do Mule cresceu proporcionalmente a cada versão, com um aumento de aproximadamente cinquenta porcento da versão 3.1 até a versão 3.8, tanto de classes e interfaces como de métodos. O que influenciou diretamente no resultado das Figuras 3 e 4. Onde na Figura 3, desde o começo a complexidade ciclomática de McCabe [5] do sistema já ultrapassava em muito o recomendado (10), chegando ao triplo e quase quatro vezes maior na versão 3.8. Ainda o peso da soma da complexidade ciclomática de McCabe [5] que está diretamente relacionada aos métodos, o quão complexos e difíceis são, teve um aumento de quase cinquenta porcento com o crescimento significativo do sistema. Demonstrando que além de o tamanho do sistema ter aumentado muito, a sua complexidade algorítmica continua elevada em cada classe, tornando o nível de complexidade do sistema elevado e dificultando em muito a manutenibilidade do software.

V. Conclusão

A integração e suas ferramentas surgem com a necessidade de integrar aplicações novas, o que exige que a ferramenta esteja em constante evolução. Porém a evolução desta ferramenta pode acabar por aumentar o seu grau de dificuldade e complexidade para uma manutenção adaptativa com os padrões de softwares recomendados. O uso das métricas possibilitam averiguar o nível de complexidade do código fonte de um software permitindo descobrir o grau de complexidade da ferramenta de integração Mulesoft, que está em constante evolução em suas versões e a um nível elevado de complexidade para a manutenibilidade adaptativa.

Referências

[1] Corchuelo R. Frantz F.R. Frantz, R.Z. On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111:89–104, 2015.

[2] Gregor Hohpe and Bobby Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[3] Anne Geraci Jane Radatz and Freny Katki. *Ieee standard glossary of software engineering terminology*. IEEE Std, 1990.

[4] Marinescu R. Lanza, M. *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer, 2006.

[5] Thomas J McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, 4:308–320, 1976.