SIMIFLOW: UMA ARQUITETURA PARA AGRUPAMENTO DE WORKFLOWS POR SIMILARIDADE

Vítor Silva Sousa

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Marta Lima de Queirós Mattoso
Eduardo Soares Ogasawara

Rio de Janeiro
Novembro de 2011

SIMIFLOW: UMA ARQUITETURA PARA AGRUPAMENTO DE WORKFLOWS POR SIMILARIDADE

Vítor Silva Sousa

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinada por:

_____

Prof. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof. Eduardo Soares Ogasawara, M.Sc.

_____

Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, D. Sc.

_____

Prof. Daniel Cardoso Moraes de Oliveira, M.Sc.

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO de 2011

Sousa, Vítor Silva

SimiFlow: Uma Arquitetura para Agrupamento de Workflows por similaridade / Vítor Silva Sousa. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2011.

VIII, 66 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso e Eduardo Soares Ogasawara

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2011.

Referências Bibliográficas: p. 64-67.

1. Arquitetura 2. Workflows Científicos 3. Linhas de Experimento 4. Clusterização I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Título.

A minha mãe Teresa Cristina, ao meu pai Carlos,

a minha irmã Marina e aos meus amigos,

por tudo que representam na minha vida.

AGRADECIMENTOS

À Marta Mattoso, minha orientadora, por ter acreditado no meu potencial desde o segundo período do curso, pelas críticas muito valiosas no amadurecimento dos trabalhos, pelas sugestões do meu encaminhamento acadêmico e pelas excelentes oportunidades oferecidas;

Ao Eduardo Ogasawara, meu orientador, que sempre me proporcionou um ambiente incomparável para aquisição de conhecimento, aplicação dos conceitos aprendidos em aula, experiências de vida e mesmo a difícil habilidade de separar o ambiente do trabalho com o da amizade;

Aos membros da banca, Alvaro Coutinho e Daniel de Oliveira, por aceitarem em participar da apresentação do meu projeto de graduação;

Ao aluno de graduação Fernando Chirigati pela amizade e pelo trabalho em equipe que desempenhamos ao longo desses anos;

À Mara Prata e Patrícia Leal por sempre me ajudarem com as questões administrativas;

A todos os amigos, que me apoiaram ao longo do curso e acreditaram no meu potencial para o desenvolvimento desse projeto. Amigos que manifestaram em pequenos atos esse sentimento tão difícil de ser mensurada, a amizade. Alguns exemplos podem ser citados, como: as caronas para Niterói da Carolina; a atenção do Jonas Dias ao explicar os conceitos e aplicações de computação; e as companhias de Fernando Pinheiro, Igor Campbell, Lívia Pimentel, Luiz Alves, Pedro Lemos, Talita Lopes e Thaiana Pinheiro;

À família Bruno, em especial à Thayana, pela atenção, apoio e contribuição para enfrentar os obstáculos encontrados ao longo do curso. Percebo o quanto a forma inovadora de propor soluções foi decisória na definição de trabalhos a serem desenvolvidos em diferentes disciplinas ou mesmo nos métodos de estudo adotados. Acredito que minha capacidade criativa foi fortemente desenvolvida pelo seu apoio, mesmo diante de críticas necessárias para o meu aperfeiçoamento;

Agradeço.

Resumo do Projeto de Graduação apresentação à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

SimiFlow: Uma Arquitetura para Agrupamento de Workflows por Similaridade

Vítor Silva Sousa

Novembro/2011

Orientadores: Marta Lima de Queirós Mattoso
Eduardo Soares Ogasawara

Curso: Engenharia de Computação e Informação

Os cientistas tem utilizado Sistemas de Gerência de *Workflows* Científicos (SGWfC) para apoiar experimentos científicos. Contudo, um SGWfC utiliza uma linguagem própria para a modelagem de um *workflow*, a ser futuramente executado. Os cientistas não possuem um auxílio ou orientação para obter o *workflow* modelado. As linhas de experimentos, que são uma nova abordagem para lidar com essas limitações, permitem uma representação abstrata e uma composição sistêmica dos experimentos. Dado que já existem muitos *workflow*s científicos previamente modelados, os cientistas podem usá-los para alavancar a construção de novas representações abstratas. Esses experimentos anteriores podem ser úteis para formar uma estrutura abstrata, se conseguirmos agrupá-los por meio de critérios de similaridade. Esse projeto propõe a SimiFlow, que é uma arquitetura para comparação baseada na similaridade e agrupamento para construir linhas de experimentos através de uma abordagem ascendente.

Palavras-chave: Arquitetura, *Workflows* Científicos, Linhas de Experimentos, Clusterização.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

SimiFlow: An Architecture for Workflow Clustering by Similarity

Vítor Silva Sousa

November/2011

Advisors: Marta Lima de Queirós Mattoso

Eduardo Soares Ogasawara

Major: Computer and Information Engineering

Scientists have been using Scientific Workflow Management Systems (SWfMS) to support scientific experiments. However, SWfMS expect a modeled workflow to be represented on its workflow language to be executed. The scientist does not have an assistance or guidance to obtain a modeled workflow. Experiment lines, which are a novel approach to deal with these limitations, allow for the abstract representation and systematic composition of experiments. Since there are many scientific workflows already modeled and successfully executed, they can be used to leverage the construction of new abstract representations. These previous experiments can be helpful by identifying scientific workflow clusters that are generated according to similarity criteria. This project proposes SimiFlow, which is an architecture for similarity-based comparison and clustering to build experiment lines following a bottom-up approach.

Keywords: Architecture, Scientific Workflows, Experiment Lines, Clustering.

SUMÁRIO

1. Introdução

O apoio computacional a experimentos científicos tem evoluído pela inviabilidade de processar e transformar grande volume de dados pelos cientistas. Nos diversos cenários possíveis de experimentos, os cientistas executam simulações para validar seus modelos ou hipóteses, o que implica, muitas vezes, na necessidade de mecanismos computacionais muito complexos. Para isso, o conceito de *workflows* científicos foi proposto, sendo que o mesmo consiste de uma abstração, permitindo a especificação dos experimentos de uma maneira estruturada, através do uso do fluxo de programas, serviços e dados para produzir o resultado final (DEELMAN *et al*., 2009).

Devido a complexidade em compor programas para modelar um *workflow* coerente (muitas invocações de programas, etapas de transformações de dados adicionais, necessidade de *loops* e pontos de decisão), observa-se que os Sistemas de Gerência de *Workflows* Científicos (SGWfC) não apóiam por completo o cientista na composição desses *workflows* (MCPHILLIPS *et al*., 2009).

Uma das propostas existentes está relacionada ao uso de diferentes níveis de abstração para apoiar a composição de *workflows* científicos (CAVALCANTI *et al*., 2005). Os níveis de abstração apresentam duas categorias: abstrata e concreta. De acordo com a definição, um *workflow* abstrato representa uma sequência de atividades, em que os recursos de execução não possuem seu mapeamento especificado. Por outro lado, os *workflows* concretos conectam a execução das atividades com os recursos computacionais, sendo geralmente especificados e executados em Sistemas de Gerência de *Workflows* Científicos (SGWfC).

Além disso, como os cientistas focam, geralmente, nos conceitos dos experimentos, houve a definição de um novo nível de abstração, sendo este conhecido por nível

conceitual. Assim como a criação de um novo nível, houve a definição de um *workflow* para o mesmo, sendo conhecido como *workflow* conceitual, cujo objetivo é encadear as atividades através da especificação do que precisa ser executado, sem determinar como deve ser implementado (OGASAWARA *et al*., 2009). Entretanto, este tipo de configuração conceitual apresenta-se como uma questão em aberto e importante, uma vez que os SGWfC não fornecem suporte à especificação dos *workflows* conceituais, restringindo-se à representação de *workflows* concretos (MATTOSO *et al*., 2010).

Sendo assim, o conceito de linhas de experimentos foi proposto para satisfazer a essas dificuldades na composição de *workflows* científicos, através de diferentes níveis de abstração (OGASAWARA *et al*., 2009). As linhas visam à minimização da complexidade de composição de experimentos científicos, definindo famílias de experimentos que compartilham atividades comuns. Desta forma, as linhas apresentam informações importantes para a composição de um experimento, como a existência de atividades opcionais e variantes (OGASAWARA *et al*., 2009), favorecendo a derivação de *workflows* científicos diferentes a partir dessas linhas.

Existem duas abordagens diferentes para modelar linhas de experimentos: descendente e ascendente. Na abordagem descendente, o cientista é capaz de compor *workflows* conceituais / abstratos e gerar *workflows* concretos a partir desses *workflows* modelados previamente, dado que o cientista já tenha a especificação necessária. Esse processo descrito é conhecido por derivação. Além disso, a necessidade de especificação de novos experimentos pelos cientistas e a inviabilidade de obter os *workflows* concretos modelados anteriormente para serem reutilizados, favorece o uso da abordagem descendente.

Por outro lado, a composição de *workflows* pode ser realizada pela abordagem ascendente. Nesse caso, os *workflows* conceituais e abstratos podem ser modelados a

partir de *workflows* concretos existentes e que são importados. Na abordagem ascendente, os cientistas buscam a identificação do *workflow* conceitual a partir de vários *workflows* concretos já modelados. Naturalmente, um único *workflow* conceitual pode ter associado a uma grande quantidade de *workflows* concretos, sendo que o cientista pode não ter ciência desse fato. Dessa maneira, *workflows* concretos pré-existentes podem ser comparados, quanto a sua estrutura, o que permite a execução da abordagem ascendente para criar linhas de experimentos. Contudo, há a necessidade de definição de heurísticas para calcular a similaridade entre os *workflows* e agrupar os mesmos baseados nesse cálculo.

O objetivo principal desse projeto reside no processo de comparação de *workflows* e agrupamento que pode ser posteriormente utilizado para modelagem das linhas de experimentos. Para isso, esse projeto apresenta a SimiFlow (SILVA *et al*., 2010) (SILVA *et al*., 2011), uma arquitetura que se concentra na comparação e agrupamento de *workflows* baseado na similaridade. A SimiFlow visa a apoiar a modelagem de linhas de experimentos seguindo uma abordagem ascendente. A arquitetura SimiFlow é dividida em três cartuchos, sendo eles, o cartucho de importação de *workflows*, de comparação por similaridade e de agrupamento de *workflows*.

O primeiro cartucho efetua a importação de *workflows* a partir de arquivos XML. Já o segundo cartucho, realiza a comparação de todos os pares de *workflows* já importados, retornando um valor de similaridade entre os *workflows*. Por último, o cartucho de agrupamento de *workflows* é responsável pela identificação de *workflows* que são similares entre si e a sua separação em grupos. Essa arquitetura foi implementada como uma extensão da ferramenta GExpLine (OLIVEIRA *et al*., 2010), que permite a composição de *workflows* por meio do conceito de linhas de experimentos (OGASAWARA *et al*., 2009).

O projeto está dividido em sete capítulos, além dessa introdução. O Capítulo 2 tem o objetivo de apresentar os conceitos de experimentos científicos. O conceito de linhas de experimentos é abordado no Capítulo 3, com o intuito de enfatizar o uso de um nível de abstração na definição de *workflows* conceituais/abstratos. No Capítulo 4, a ferramenta GExpLine é apresentada para o apoio das linhas de experimentos. O Capítulo 5 trata da arquitetura SimiFlow, descrevendo seus cartuchos, enquanto que o Capítulo 6 revela a avaliação experimental realizada. E por último, o Capítulo 7 conclui o trabalho.

2. Experimento Científico

Esse capítulo tem o propósito de apresentar os principais conceitos sobre experimentos científicos. Para isso, o Subcapítulo 2.1 descreve o ciclo de vida de um experimento conforme proposto em (MATTOSO *et al*., 2009). Já o Subcapítulo 2.2 apresenta *workflows* científicos, o seu papel no ciclo de vida e os principais componentes necessários na definição dos mesmos. O Subcapítulo 2.3, por sua vez, descreve o apoio e a definição de Sistemas de Gerência de *Workflows* Científicos para atender ao ciclo de vida de um experimento. Por último, tendo como intuito facilitar o desenvolvimento de *workflows* pelos cientistas, diferentes níveis de abstração são apresentados no Subcapítulo 2.4.

2.1. Ciclo de Vida de um Experimento Científico

Devido à necessidade de organizar, processar, controlar e analisar os experimentos para atender aos modelos propostos pelos cientistas, o conceito de ciclo de vida de um experimento científico foi proposto (MATTOSO *et al*., 2009). Através desse ciclo é possível definir os experimentos, assim como a execução permite avaliar se os resultados obtidos estão de acordo com as definições dos próprios. Caso não estejam, os experimentos podem ser alterados. Essas etapas são conhecidas como ciclo de vida de um experimento científico e é dividido em três fases: composição, execução e análise (MATTOSO *et al*., 2009).

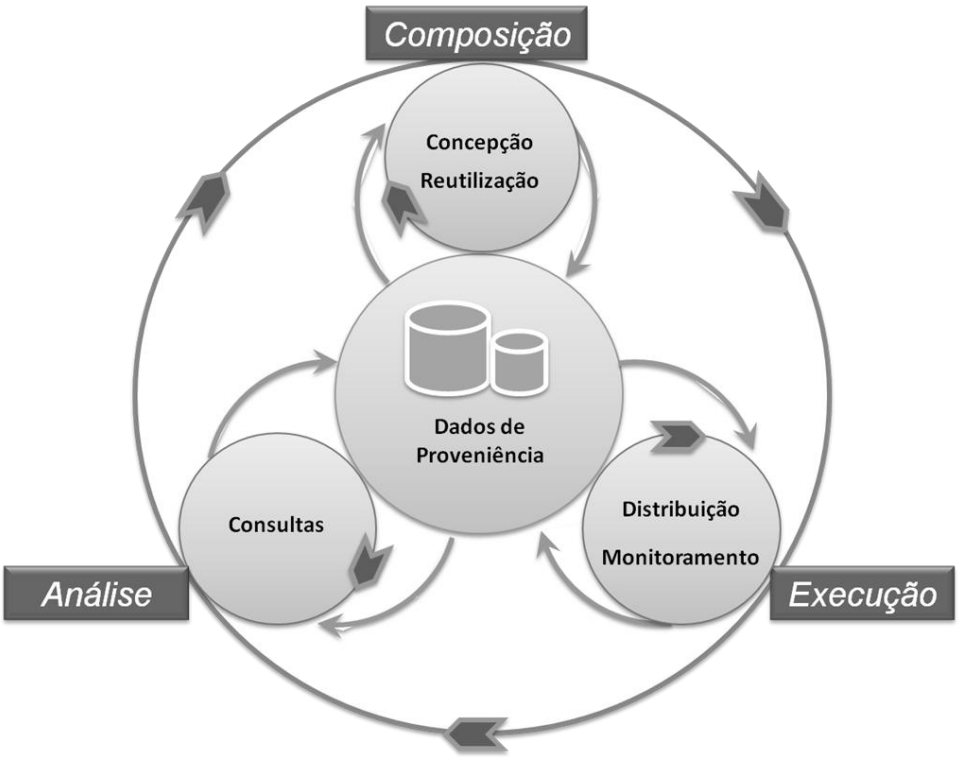

Figura 1. Ciclo de vida do experimento científico adaptado de (MATTOSO *et al*., 2009).

Conforme o ciclo de vida do experimento apresentado na Figura 1, a fase de composição consiste na definição dos programas, que devem fazer parte do *workflow* visando atender ao experimento científico em questão. Nessa fase, podem-se criar novas versões desses programas ou reutilizar outros programas presentes em experimentos anteriores.

Já a fase de execução caracteriza-se pelo processamento propriamente dito dos *workflows* desenvolvidos na etapa anterior, assim como o seu monitoramento. Pelo fato dos *workflows* necessitarem de alto recurso computacional e manipulação de grande volume de dados, a proposta de soluções que utilizem alto desempenho computacional e de computação paralela é vantajosa, caso haja disponibilidade desses recursos.

Por último, a fase de análise consiste na avaliação dos resultados obtidos pela execução dos *workflows* científicos. Um mecanismo existente para analisar os resultados obtidos é a proveniência dos dados, cujo objetivo é permitir uma avaliação do experimento que pode conter informações tanto da fase de composição quanto da fase de execução. Desta forma, a proveniência dos dados pode ser dividida em dois tipos: a prospectiva e a retrospectiva (FREIRE *et al*., 2008). A proveniência prospectiva está associada à captura das informações durante a fase de composição dos experimentos científicos. Enquanto isso, a proveniência retrospectiva diz respeito à coleta de informações em tempo de execução do *workflow*.

2.2. *Workflow* Científico

Os experimentos científicos necessitam, geralmente, da execução de uma grande quantidade de simulações computacionais a fim de avaliar seus modelos ou hipóteses. Os *workflows* científicos são caracterizados por fornecer a abstração que permite a especificação desses experimentos de maneira estruturada, por meio de um fluxo de programas, serviços e dados para produzir um resultado final (DEELMAN *et al*., 2009). Tal resultado tem o objetivo de atender as necessidades do experimento em questão.

O fluxo de programas, serviços e dados pode ser definido a partir dos componentes constituintes de um *workflow* científico, sendo os seguintes: atividade, porta, relacionamento e *sub-workflow*. A atividade é responsável por consumir e produzir dados, sendo assim caracterizada pela execução de um programa ou serviço.

Já a porta especifica quais dados devem ser consumidos e produzidos em cada atividade. Além disso, cada definição de porta contém a estrutura de dados utilizada, como por exemplo, uma porta do tipo inteiro. Outra característica importante na

especificação das portas de uma atividade consiste em avaliar se uma determinada porta comporta-se apenas para o consumo de dados (porta de entrada), apenas para a produção de dados (porta de saída) ou tanto para o consumo, como para a produção de dados (porta de entrada e saída). Para as portas de entrada e de saída, o dado de entrada será o mesmo que o dado produzido pela atividade em questão.

Um relacionamento é o componente que permite o encadeamento lógico das atividades, ou seja, define como é o fluxo de dados ou de controle entre as atividades. Portanto, ao especificar um relacionamento num *workflow* científico é necessário estabelecer qual é o fluxo dos dados, ou seja, definem-se as portas de origem e de destino do relacionamento. Ao mesmo tempo em que as portas são estabelecidas, as atividades também são, apresentando qual atividade produz um dado e, qual atividade consumirá esse dado. Sendo assim, a porta de origem é atribuída à porta de saída de uma atividade, enquanto que a porta de destino é atribuída a uma porta de entrada de outra atividade.

A Figura 2 apresenta um esquema geral de um *workflow* científico, em que é possível observar a existência de duas atividades (atividades *A* e *B*). A atividade *A* apresenta três portas de entrada (portas *X*, *Y* e *Z*) e duas portas de saídas (portas *a* e *b*). Enquanto isso, a atividade *B* apresenta duas portas de entrada (portas *W* e *d*) e duas portas de saída (portas *c* e *d*). Vale ressaltar que o nome igual para as duas portas (porta *d*), tanto para a entrada como para saída, enfatiza a presença de uma porta do tipo entrada e saída. Além das atividades e das portas, observam-se os relacionamentos presentes entre as atividades *A* e *B* (relacionamentos *rel1* e *rel2*). Para cada relacionamento existe a especificação da atividade de origem, porta de origem, atividade de destino e porta de destino. No caso do relacionamento *rel1*, a porta *a* é a porta de origem e a porta *W* é a porta de destino. Dessa forma, para o relacionamento

rel1, as atividades de origem e de destino são, respectivamente, *A* e *B*. Assim como ocorre para o relacionamento *rel1*, o relacionamento *rel2* apresenta as mesmas atividades de origem e de destino. Contudo as portas de origem e de destino são diferentes, sendo, respectivamente, as portas *b e d*.

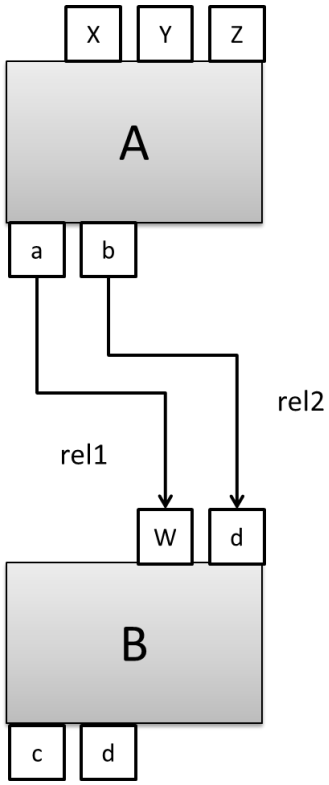

Figura 2. Esquema geral de um *workflow* científico, sem considerar a existência de *sub-workflows*.

Já um *sub-workflow* é caracterizado pela existência de um *workflow* completo dentro da especificação de uma atividade, logo, esse componente consiste na existência de um *workflow* dentro de outro *workflow*. A Figura 3 apresenta um exemplo de *workflow* com uma atividade (atividade *A*), em que essa atividade comporta-se como um *sub-workflow*, composto de três atividades (atividades A_1, A_2 e A_3).

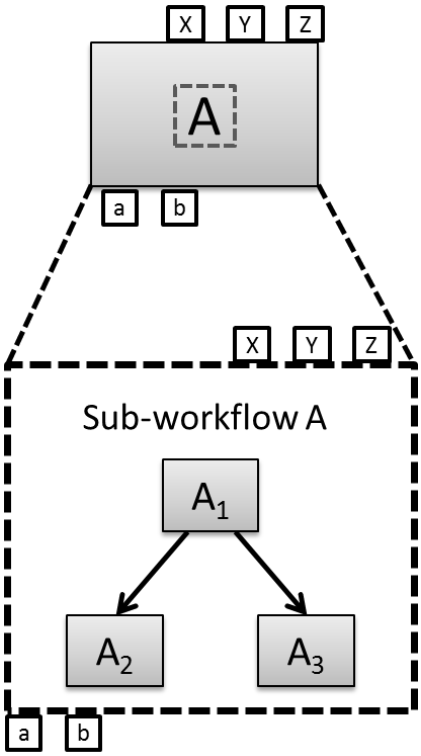

Figura 3. Exemplo de um *sub-workflow* em uma atividade.

2.3. Sistemas de Gerência de *Workflows* Científicos

Com o intuito de permitir a composição, execução e monitoramento de *workflows* científicos, os Sistemas de Gerência de *Workflows* Científicos (SGWfC) foram desenvolvidos. Através dos mesmos define-se a execução dos programas e serviços, assim como a manipulação dos dados existentes num *workflow*, de acordo com o experimento científico que se deseja representar. Existem diversos SGWfC diferentes, como Kepler (ALTINTAS *et al*., 2004), VisTrails (CALLAHAN *et al*., 2006) e Taverna (OINN *et al*., 2004), sendo que cada um apresenta suas próprias características e comportamentos.

O Kepler (ALTINTAS *et al*., 2004) (LUDÄSCHER *et al*., 2006) é um sistema de gerência de *workflow* de código aberto, originado no projeto Ptolemy II. O seu principal

objetivo é o desenvolvimento de experimentos nas áreas de geoinformática, bioinformática e ecologia, através do suporte à composição, execução e análise de *workflows* científicos. A ferramenta oferece uma interface de fácil uso, uma vez que as principais operações concentram-se na área de edição do *workflow*, que pode ter elementos adicionados através de ações *drag-and-drop*. A Figura 4 apresenta um exemplo de *workflow* desenvolvido no Kepler.

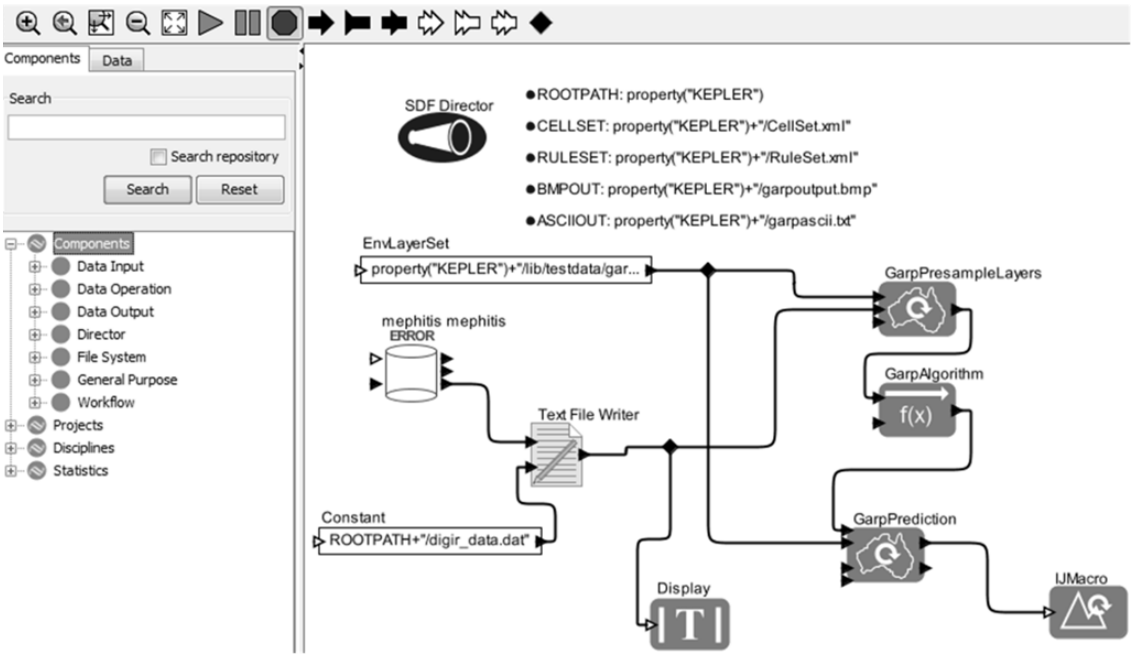

Figura 4. Exemplo de *workflow* modelado no Kepler.

As atividades do SGWfC Kepler são denominadas de atores, sendo que existem diferentes atividades pré-definidas nesse sistema para desempenhar uma determinada funcionalidade. Já os relacionamentos são representados pelo diretor, que define as dependências das relações. Essa dependência é responsável por controlar o momento, em que cada atividade deve ser executada.

Para diminuir a complexidade de composição de *workflows*, o Kepler também propôs um mecanismo de suporte a *sub-workflows* (TAYLOR *et al*., 2007) no nível concreto. Todavia, o uso de níveis superiores de abstração não é oferecido por essa ferramenta.

O VisTrails é outro SGWfC, cujo principal propósito está associado a captura e armazenamento de dados de proveniência dos *workflows*. Para isso, o mesmo apresenta um conjunto de funcionalidades que proporciona o controle de versões ao longo do processo de composição dos *workflows* e um sistema de rastreamento das execuções. Além disso, o VisTrails apresenta uma linguagem (*VisTrails Query Language*) (ANDERSON *et al*., 2007), que proporciona o monitoramento da execução dos *workflows* através de consultas, apesar de não apresentar uma interface que facilite sua construção.

Muitas aplicações de visualização científica são desenvolvidas no SGWfC VisTrails (CALLAHAN *et al*., 2006), sendo um dos diferenciais o uso de uma área de visualização (*spreadsheet*), em que é possível visualizar e comparar os resultados das execuções de diferentes versões de um mesmo *workflow*. Em cada célula dessa planilha é apresentada uma instância da execução do *workflow*, conforme apresentado pela Figura 5.

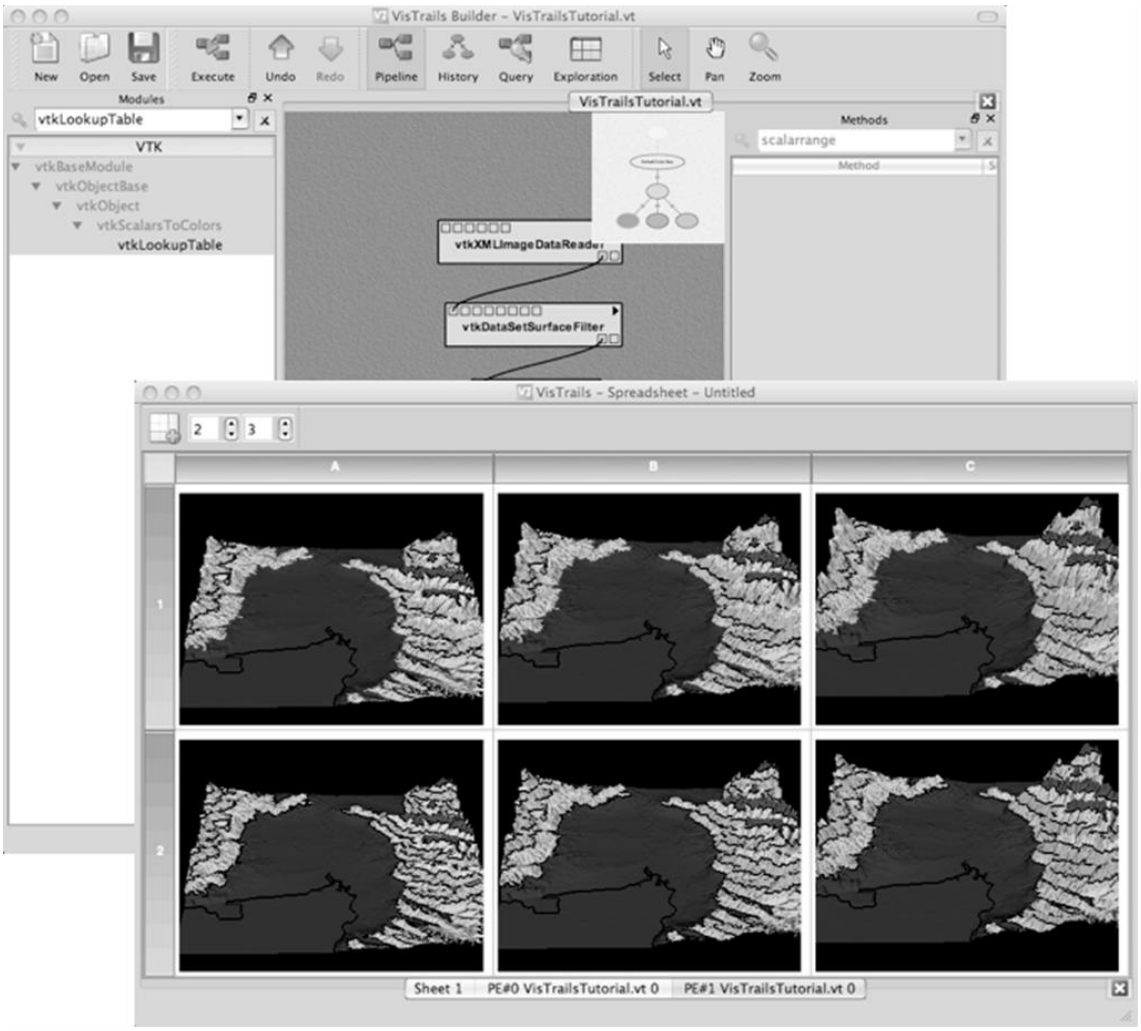

Figura 5. Aplicação de visualização científica no VisTrails.

Por último, o Taverna Workbench (OINN *et al*., 2004) (OINN *et al*., 2007) (HULL *et al*., 2006) é também um SGWfC que auxilia os usuários na composição, execução e análise de *workflows* científicos. Para apoiar a busca por serviços disponibilizados no sistema, a ontologia *myGrid* (GOBLE *et al*., 2003) (WOLSTENCROFT *et al*., 2007) (STEVENS *et al*., 2004) foi criada e subdividida em ontologia de domínio e de serviços.

A ontologia de domínio lida com os conceitos envolvidos na definição de *workflows* de bioinformática. Enquanto isso, a ontologia de serviços tem o propósito de modelar os serviços web e os parâmetros no Taverna. Através dessa ontologia é

possível obter atividades utilizadas por terceiros, apesar de não possuir os algoritmos e programas utilizados por essas atividades. A Figura 6 apresenta um exemplo de *workflow* modelado no Taverna.

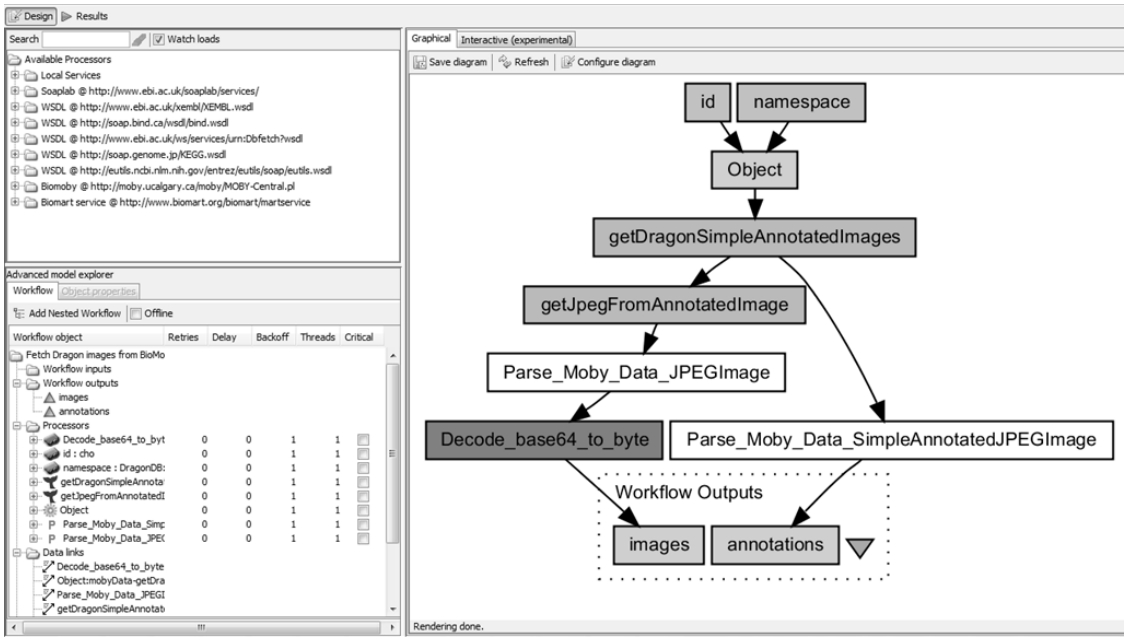

Figura 6. Exemplo de *workflow* desenvolvido no Taverna.

Também desenvolvido no projeto *myGrid*, o sítio *myExperiment* (GOBLE E ROURE, 2007) (DE ROURE E GOBLE, 2007) tem o propósito de facilitar a troca de informações entre especialistas na composição de *workflows*. Para atender a essa proposta, o *myExperiment* oferece um conjunto de mecanismos que favorecem a comunicação em relação às especificações dos *workflows*, os *logs* de proveniência ou mesmo descrições semânticas. Sendo assim, esse ambiente favorece a difusão e documentação do conhecimento para a modelagem de *workflows* em áreas determinadas pelos experimentos, em que grupos de pessoas propõem-se a desenvolver, diminuindo o tempo necessário para a composição desses experimentos individualmente.

2.4. Níveis de Abstração

Além dos componentes apresentados para a definição de *workflows* científicos, dois níveis de abstração podem ser definidos para os mesmos: concreto e abstrato. O nível abstrato lida com uma camada mais alta de abstração, sem definir propriedades muito técnicas a serem utilizadas na execução. Já o *workflow* concreto apresenta propriedades mais específicas, através da descrição dos recursos computacionais disponíveis.

Além dos dois níveis de abstração apresentados, um nível diferente foi desenvolvido para auxiliar os cientistas na definição dos conceitos envolvidos num experimento científico. Esta nova camada de abstração é conhecida também como nível conceitual. Associado a esse nível, houve a definição do *workflow* conceitual, cujo objetivo é encadear as atividades através da especificação do que precisa ser feito sem determinar como isso tem que ser feito (OGASAWARA *et al*., 2009). Entretanto, este tipo de especificação abstrata apresenta-se como uma questão em aberto e importante, uma vez que os SGWfC não fornecem suporte à especificação dos *workflows* conceituais (MATTOSO *et al*., 2010).

A Figura 7 exemplifica os três níveis de abstração: concreto, abstrato e conceitual. Nesse caso, o *workflow* conceitual (que contém apenas a atividade *A*) é representado através de duas atividades abstratas *A1* e *A2*, separando os métodos, algoritmos e técnicas que serão utilizados, através dessas duas atividades. Por outro lado, o *workflow* concreto descreverá o mecanismo físico utilizado para aplicar o método determinado no nível abstrato. O exemplo em questão apresentou o caso de um *workflow* concreto desenvolvido no SGWfC VisTrails.

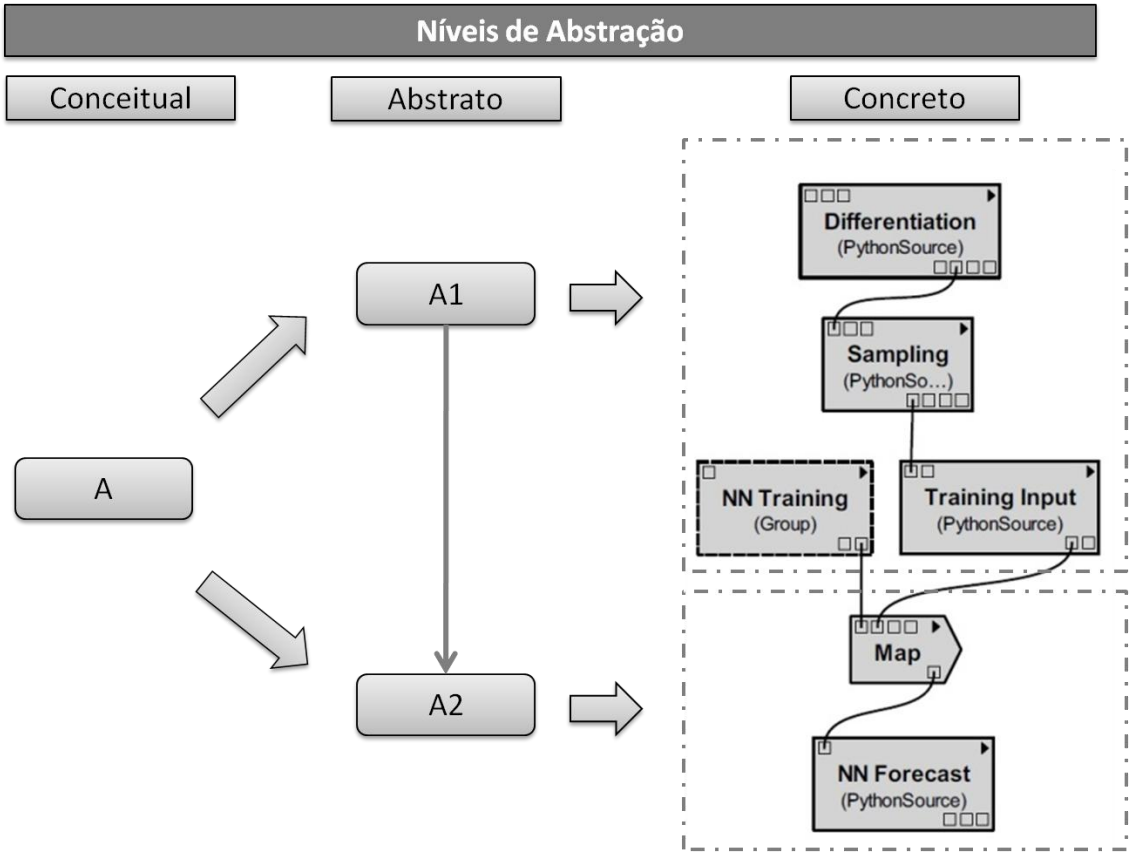

Figura 7. Exemplo de *workflows* nos três níveis de abstração: concreto, abstrato e conceitual.

A existência de níveis de abstração superiores ao nível concreto é importante para auxiliar o trabalho de cientistas na composição de *workflows* científicos, pois, ao compor um experimento através das atividades abstratas, as especificações em termos de recursos computacionais não seriam necessárias, uma vez que essas configurações estariam presentes nas atividades concretas que deram origem a definição de uma atividade abstrata, por exemplo. Assim, o trabalho dos cientistas é facilitado, apesar de ser necessário um mínimo de conhecimento quanto à finalidade de cada atividade abstrata e os parâmetros de configuração das portas existentes nessas atividades.

Além disso, os diferentes níveis de abstração podem proporcionar abordagens que facilitem a representação dos *workflows* científicos. No próximo capítulo, o conceito de

linhas de experimentos é apresentado como uma abordagem para representar *workflows* conceituais, sendo que estes podem obter *workflows* abstratos, apresentando alterações na sua estrutura quanto ao algoritmo utilizado para execução de uma atividade, por exemplo.

3. Linhas de Experimentos

Linhas de experimentos é uma adaptação do conceito de linhas de produto de software (NORTHROP, 2002), também conhecidas por SPL (do inglês, *Software Product Lines*). As SPL foram criadas para apoiar o processo de desenvolvimento de software, proporcionando métodos, ferramentas e técnicas de engenharia de software usadas para criar coleções de sistemas de software similares a partir de recursos de software que usam meios de produção comuns (NORTHROP, 2002).

A adaptação de conceitos de SPL para o contexto de *workflows* científicos, proposta por OGASAWARA *et al*. (2009), requer a compreensão de que uma atividade de *workflow* pode ser mapeada num componente de software, sendo que um componente é uma unidade de composição que estabelece dependências e uma interface bem definida (FUSARO *et al*., 1998). Além disso, existem estudos em engenharia de software que consideram os processos como componentes (FUSARO *et al*., 1998). O propósito das linhas de experimentos é estender esse conceito para *workflows* científicos.

A abordagem de linhas de experimentos apóia a fase de composição de um experimento científico, através do uso de uma representação independente. As linhas de experimentos comportam-se de acordo com o encadeamento de atividades conceituais, o que significa que as mesmas podem ser vistas como *workflows* conceituais. Além disso, as linhas contêm as definições de variabilidade e opcionalidade, que estão relacionados à atividade conceitual. É importante considerar também que cada atividade se comportará como um componente no processo de experimentação.

O conceito de variabilidade é representado por pontos de variação, sendo que estes dizem respeito ao mapeamento de uma atividade conceitual em mais de uma atividade

abstrata, ou seja, essa característica determina a dimensão de variabilidade existente nas linhas de experimentos. Portanto, uma atividade conceitual será um ponto de variação, caso exista mais de um método, algoritmo ou programa alternativo que execute o comportamento definido pela atividade.

Do mesmo modo, uma atividade conceitual é opcional, se a mesma pode ser removida de um dos mapeamentos de atividades abstratas, dado um experimento científico particular. Em contrapartida, uma atividade obrigatória consiste numa atividade que precisa estar presente no experimento científico. Portanto, essas características propiciam, para as linhas de experimentos, informações da dimensão de opcionalidade. Vale enfatizar também que as dependências entre as atividades são estabelecidas pelos relacionamentos, através da especificação de conexões entre portas de entrada e de saída das atividades.

Enquanto isso, o processo de derivação relaciona-se à obtenção de *workflows* abstratos a partir de linhas de experimentos (*workflow* conceitual). Como o comportamento de um componente apresentado pelas atividades é importante para mapear as atividades de um *workflow* conceitual em uma lista de atividades concretas ou abstratas, o processo de derivação pode ocorrer, caso as etapas seguintes sejam efetuadas:

- Escolha das atividades variantes para cada ponto de variação;
- Escolhas de quais atividades opcionais precisam ser incluídas.

A Figura 8 apresenta um exemplo de linhas de experimentos, em que dois *workflows* abstratos são derivados. Nesse caso, as atividades *A*, *B* e *E* são atividades conceituais obrigatórias, enquanto que a atividade conceitual *D* é opcional. Assim, a atividade abstrata *d* é incluída no *workflow* abstrato derivado #1, enquanto que a mesma não faz parte do *workflow* abstrato derivado #2. Além disso, a atividade *C* é um ponto

de variação, sendo que *c1*, *c2* e *c3* são atividades variantes de *C*. Portanto, os *workflows* derivados são obtidos através da escolha das atividades variantes de cada ponto de variação, mas também considerando as atividades opcionais. É importante enfatizar que existem casos em que as atividades podem ser opcionais e pontos de variação ao mesmo tempo.

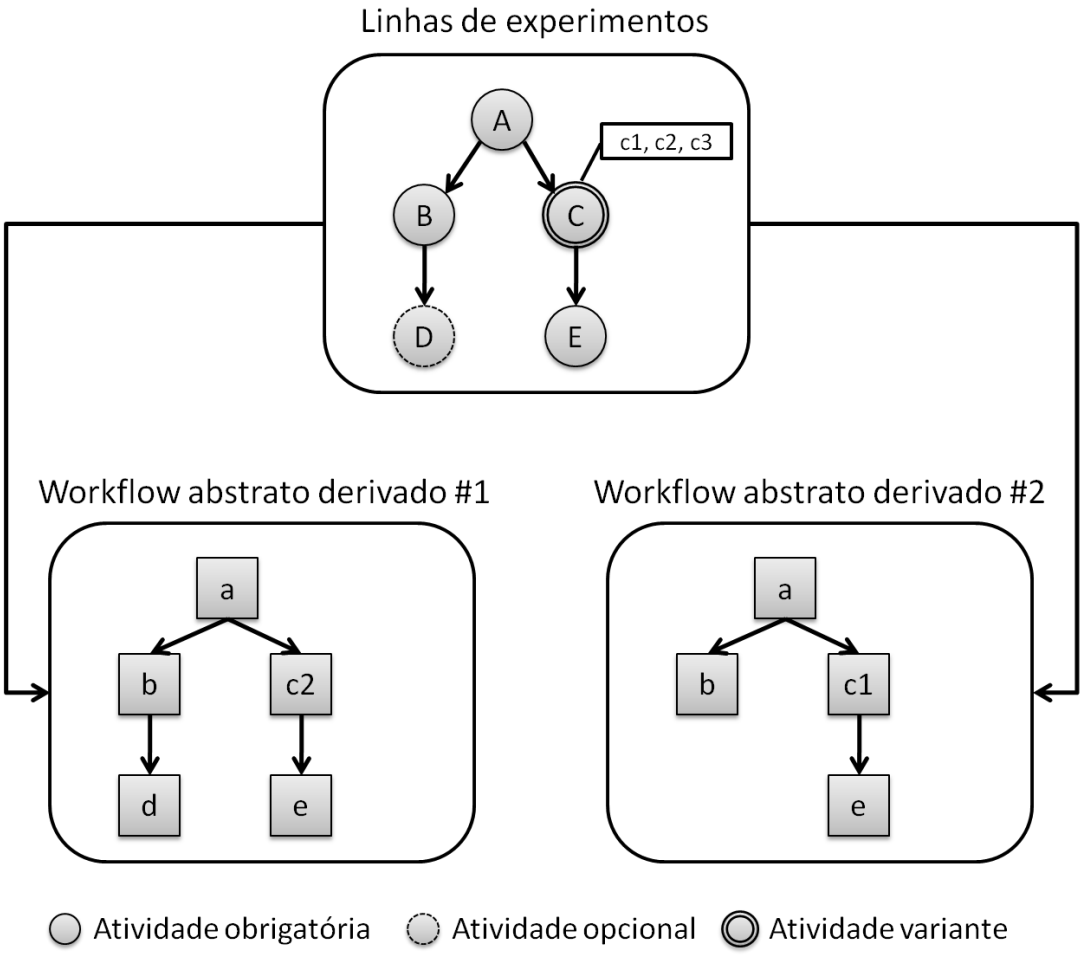

Figura 8. Exemplo de *workflows* abstratos derivados das linhas de experimentos.

Durante a etapa de composição, é importante verificar se os *workflows* derivados obtidos a partir das linhas de experimentos são válidos, ou seja, se atendem ao experimento científico proposto. Para isso, regras de composição são necessárias para realizar esse processo de checagem (GARG *et al*., 2003). Por exemplo, pode-se supor

como regra de composição, que caso a atividade abstrata *c2* seja escolhida para o *workflow* derivado, a atividade abstrata *d* também deverá estar presente no *workflow* derivado. Portanto, as regras de composição são geralmente usadas para definir as combinações possíveis de programas derivados a partir das linhas de experimentos. Esse conceito de regras de composição foi inspirado em SPL.

O capítulo a seguir tem o propósito de apresentar a ferramenta GExpLine, sendo esta um exemplo de aplicação de linhas de experimentos na composição de *workflows*.

4. GExpLine – Gerência de Linhas de Experimentos

Com o propósito de apoiar a composição de *workflows* usando linhas de experimentos, a ferramenta GExpLine foi desenvolvida (OLIVEIRA *et al*., 2010). O seu principal objetivo consiste em prover um ambiente para os cientistas modelarem seus experimentos científicos no nível conceitual. Ao mesmo tempo, a GExpLine também é responsável pela derivação de *workflows* conceituais em *workflows* concretos para um conjunto de SGWfC existentes, obedecendo as suas linguagens de especificação. A Figura 9 apresenta um exemplo de *workflow* conceitual sendo derivado para um *workflow* concreto no SGWfC VisTrails.

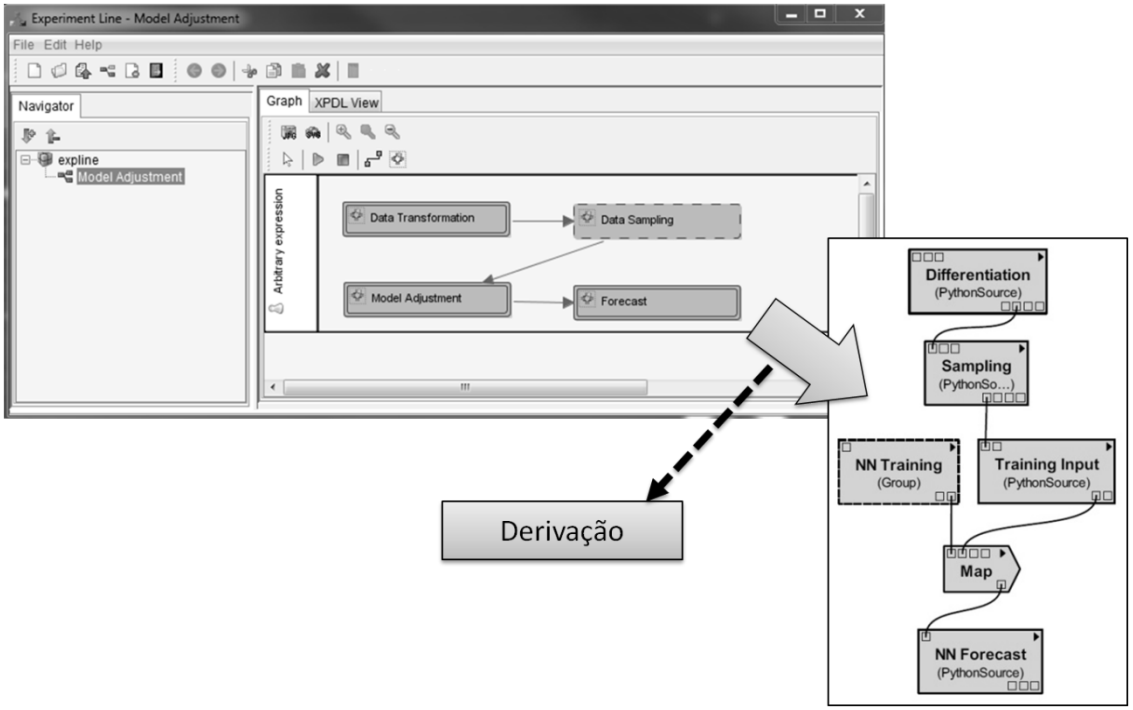

Figura 9. Processo de derivação da ferramenta GExpLine para o VisTrails.

A ferramenta GExpLine foi desenvolvida usando a linguagem de programação Java. Utilizou-se também a ferramenta de código aberto Enhydra JaWe (Java *Workflow*

Editor) (JWE, 2009), que oferece o formato XPDL (*XML Process Definition Language*) (WfMC, 2009), sendo este um formato para a representação gráfica de *workflows* e processos. Vale ressaltar que a padronização foi realizada pelo consórcio *Workflow Management Coalition* (WfMC, 2009), contudo algumas adaptações foram necessárias para atender aos conceitos de linhas de experimentos, como a variabilidade e a opcionalidade.

As definições dos *workflows* concretos e abstratos compostos são armazenadas em um banco de dados relacional. O Sistema de Gerência de Banco de Dados (SGBD) utilizado para a definição desse esquema foi o PostgreSQL.

A ferramenta GExpLine é baseada em quatro componentes principais, sendo eles:

- Modelador de linhas de experimentos, que é responsável pelo apoio ao projeto das linhas de experimentos;
- Derivador, que transforma os *workflows* conceituais, modelados pelo conceito de linhas de experimentos, em *workflows* concretos, que são capazes de executar em SGWfC específicos. No caso, a GExpLine apóia a derivação para os SGWfC Kepler (ALTINTAS *et al*., 2004), Taverna (OINN *et al*., 2004) e VisTrails (CALLAHAN *et al*., 2006);
- Controlador de versão, que controla a evolução dos *workflows* concretos, abstratos e conceituais;
- Interface de consulta, tendo como objetivo a consulta da proveniência prospectiva, utilizando informações abstratas ou concretas. As consultas de proveniência, na versão atual, são executadas diretamente em SQL a partir das funcionalidades do SGBD.

A Figura 10 apresenta os elementos que permitem a definição das linhas de experimentos na GExpLine. A classe *Workflow* apresenta um *workflow* conceitual,

abstrato ou concreto, sendo representado pelas classes *ConceptualWorkflow*, *AbstractWorkflow* ou *ConcreteWorkflow*, respectivamente.

Um *workflow* é composto por atividades (classe *Activity*) e relacionamentos (classe *Relationship*). As atividades podem ser conceituais (classe *ConceptualActivity*), abstratas (classe *AbstractActivity*) ou concretas (classe *ConcreteActivity*). Cada atividade também possuirá um conjunto de portas (classe *Port*). Uma atividade conceitual é representada por um nome e uma descrição do conceito. Uma atividade abstrata, por sua vez, é representada pelo nome do programa e algum metadado que descreva sua funcionalidade. Já uma atividade concreta é uma instanciação particular de uma atividade abstrata que contém o código necessário para executar o programa em um SGWfC específico.

Além disso, as atividades podem estar conectadas através de relacionamentos diretos (classe *Relationship*) a partir da especificação das portas. Embora não seja representado no diagrama por ser simples, *workflows* conceituais só podem ter atividades conceituais, assim como *workflows* abstratos e concretos só podem ter atividades abstratas e concretas, respectivamente.

As atividades conceituais podem estar também relacionadas com uma ou mais atividades abstratas. Se uma atividade conceitual é relacionada a mais de uma atividade abstrata, ela é chamada de ponto de variação (classe *VariationPoint*). Por outro lado, se uma atividade conceitual puder ser suprimida durante a geração de uma ou mais atividade concretas, ela é chamada de elemento opcional (classe *OptionalElement*). Do mesmo modo, a partir de um *workflow* abstrato selecionado, é possível obter diretamente os *workflows* concretos correspondentes. Pode-se enfatizar também que a Figura 10 apresenta as classes em branco para representar os conceitos gerais de

workflow, enquanto que as classes em cinza representam os conceitos de linhas de experimentos.

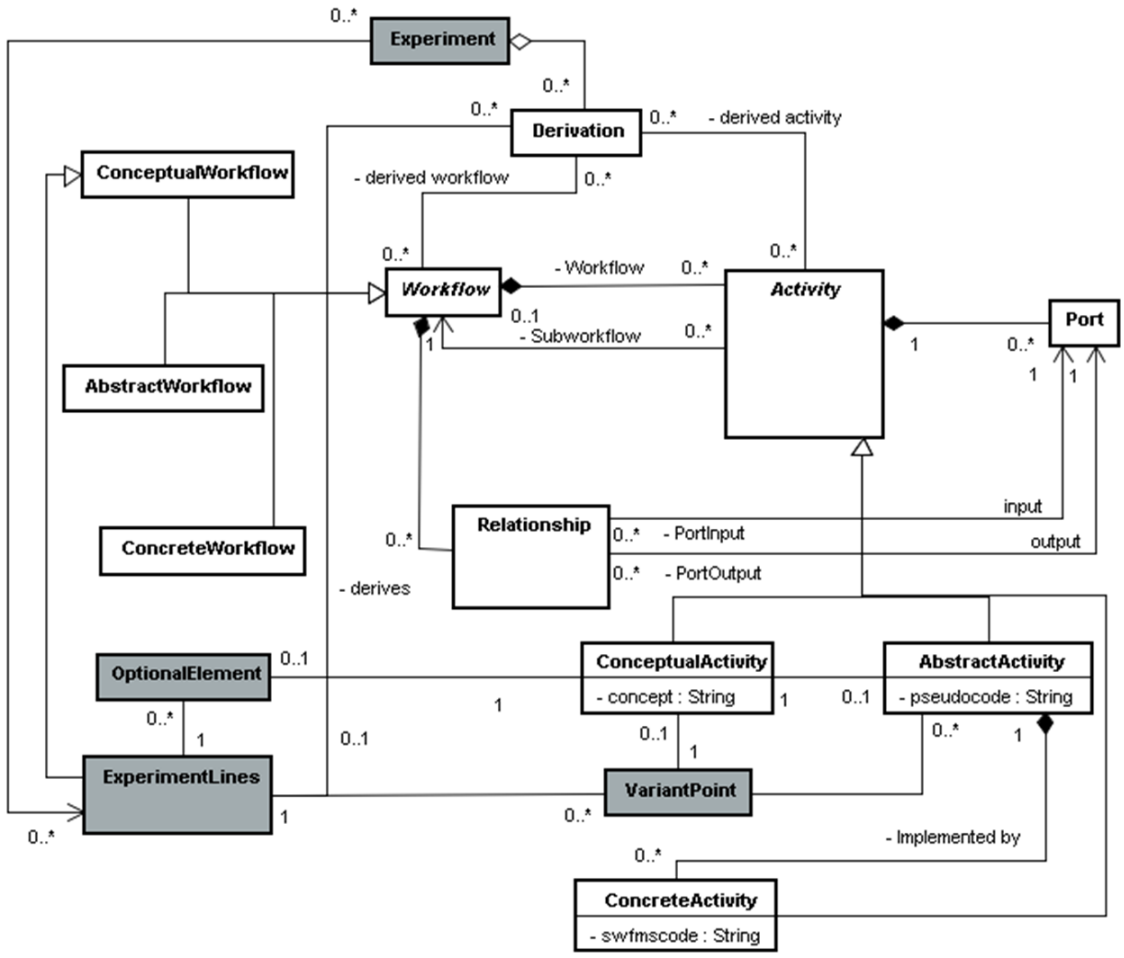

Figura 10. Diagrama de classes adaptado por (COSTA *et al*., 2009). Elementos para definição de linhas de experimentos na GExpLine.

A ferramenta GExpLine oferece uma região de edição dos *workflows*, sendo essa região conhecida por área de trabalho (*workspace*). O processo de *check-out* é o responsável por proporcionar a edição dos *workflows*, uma vez que o mesmo cria cópias de artefatos para serem utilizados na *workspace* (CONRADI E WESTFECHTEL, 1998). Além da possibilidade de edição, há a capacidade de armazenar as mudanças

realizadas através do processo de *check-in*, através da escrita das mudanças no repositório de origem.

Sendo assim, a ferramenta oferece os seguintes recursos:

- Controle de acesso e controle de concorrência sobre o repositório de *workflows*, permitindo o armazenamento e recuperação de estável e desenvolvimento de versões de *workflows*;
- Registro de versões de atividades que estão sendo usadas para cada composição de *workflows*;
- Fornecimento de uma área de trabalho para apoio na edição de *workflows*; a área de trabalho também apóia a publicação de versões de *workflows* para o repositório.

5. SimiFlow

O propósito desse capítulo é apresentar a arquitetura SimiFlow (SILVA *et al*., 2011). Para isso, o mesmo foi dividido em quatro subcapítulos. O Subcapítulo 5.1 tem o objetivo de definir a SimiFlow, bem como a estruturação de sua arquitetura baseada em cartuchos (BIRSAN, 2005). Enquanto isso, o Subcapítulo 5.2 apresenta o cartucho de importação de *workflows*, que consiste na obtenção da estrutura de *workflows* a partir de *workflows* já modelados em SGWfC. O Subcapítulo 5.3 descreve, por sua vez, o cartucho de comparação por similaridade. Nesse caso, todos os *workflows* importados pelo primeiro cartucho são comparados entre si, tendo como produto dessa operação, um valor de similaridade (para cada par de *workflows*), que consiste no quanto esses *workflows* são semelhantes. Já o Subcapítulo 5.4 apresenta o cartucho de agrupamento, que consiste na determinação de grupos de *workflows* similares. E, por último, o Subcapítulo 5.5 apresenta os trabalhos relacionados.

5.1. Arquitetura

A arquitetura SimiFlow apresenta como principal objetivo o agrupamento de *workflows* semelhantes quanto a sua estrutura. Em outras palavras, a SimiFlow busca agrupar *workflows* concretos semelhantes em relação a sua estrutura, de forma a criar linhas de experimentos através de uma abordagem ascendente que, por sua vez, poderão ser derivadas nesse *workflows* concretos que lhe deram origem e em outros (SILVA *et al*., 2011).

Para que seja possível determinar o quanto os *workflows* são semelhantes entre si, diferentes tipos de algoritmos para o cálculo de similaridade podem ser definidos. Em função dessa necessidade, a SimiFlow apresenta uma arquitetura expansível com o intuito de apoiar a especificação de diversos algoritmos e métodos tanto para a comparação, como para agrupamento dos *workflows*, através do padrão de design *strategy* (GAMMA *et al*., 1994). A existência de diferentes algoritmos é permitida pelo padrão *strategy*, pois o mesmo realiza o encapsulamento desses algoritmos em objetos.

A arquitetura SimiFlow foi desenvolvida na ferramenta GExpLine (OLIVEIRA *et al*., 2010), cujo objetivo é apoiar a composição de *workflows* através do conceito de linhas de experimentos, conforme apresentado no Capítulo 4. Essa ferramenta permite que os cientistas componham seus experimentos científicos no nível conceitual e derivem diferentes *workflows* concretos a partir das linhas de experimentos criadas, através da definição das atividades variantes a serem utilizadas e da escolha das atividades opcionais que estarão presentes no *workflow* derivado.

A SimiFlow pode ser organizada em três principais processos, conforme apresentado pela Figura 11. O primeira processo consiste na obtenção dos *workflows* modelados em SGWfC. A partir do armazenamento das características desses *workflows*, o segundo processo realizaria o cálculo de similaridade em combinações dois a dois desses *workflows*. Finalmente, o terceiro processo realiza o agrupamento dos *workflows* em famílias de *workflows* semelhantes, utilizando o valor de similaridade como métrica de separação dos *workflows* em grupos.

Figura 11. Processos realizados pela arquitetura SimiFlow.

A partir dos processos apresentados, a arquitetura SimiFlow foi estruturada em três cartuchos diferentes (BIRSAN, 2005), em que cada um busca atender a um processo específico da Figura 11. Os três cartuchos são os seguintes: cartucho de importação de *workflows*, cartucho de comparação por similaridade e cartucho de agrupamento. A Figura 12 apresenta um esquema da arquitetura.

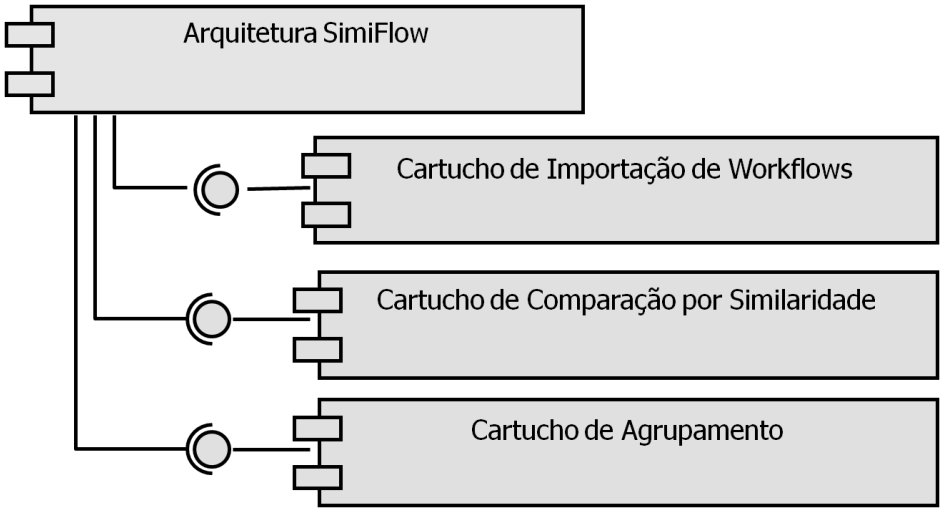

Figura 12. Arquitetura SimiFlow.

O primeiro cartucho (cartucho de importação de *workflows*) tem o propósito de obter as especificações de *workflows* concretos modelados em SGWfC e o armazenamento das estruturas desses *workflows* no esquema da base de dados da ferramenta GExpLine. Já o segundo cartucho (cartucho de comparação por similaridade) é responsável pela comparação par a par dos *workflows* obtidos pelo cartucho de importação de *workflows*, tendo como resultado um valor de similaridade calculado, que expressa o quanto dois *workflows* são semelhantes. Por último, o cartucho de agrupamento realiza alocação dos *workflows* em grupos tendo como métrica o valor de similaridade calculado pelo segundo cartucho.

É importante enfatizar também que a arquitetura SimiFlow apresenta como limitação o fato da mesma não realizar a criação de linhas de experimentos. Logo, apesar do cartucho de agrupamento determinar a que grupo cada *workflow* pertence, o mesmo não realiza o desenvolvimento de linhas de experimentos a partir desses grupos obtidos.

5.2. Cartucho de Importação de *Workflows*

O cartucho de importação de *workflows* é responsável pela obtenção e armazenamento de informações importantes quanto à estrutura dos *workflows* num esquema de base de dados. Como a SimiFlow foi implementada na ferramenta GExpLine (OLIVEIRA *et al*., 2010), essa arquitetura utiliza o esquema da base de dados relacional da GExpLine para realizar o armazenamento (SILVA *et al*., 2011).

O processo de obtenção das informações estruturais dos *workflows* é realizado através da leitura de arquivos no formato XML, que são produzidos pelos SGWfC após o desenvolvimento de *workflows* concretos pelos cientistas. Os SGWfC que a SimiFlow apóia para importação de *workflows* são Taverna (OINN *et al*., 2004), Kepler (ALTINTAS *et al*., 2004) e VisTrails (CALLAHAN *et al*., 2006).

Após a obtenção das características dos *workflows*, as mesmas são analisadas pela SimiFlow, sendo que esta realiza o mapeamento necessário das informações existentes no arquivo XML para o esquema de base de dados relacional da GExpLine. Sendo assim, a análise dos arquivos XML consiste na determinação de que componentes do *workflow* (atividade, porta, relacionamento) cada elemento XML faz referência. Pode acontecer também de um elemento do arquivo XML descrever algumas características do próprio *workflow* (sem serem características relacionadas aos seus componentes).

A Figura 13 apresenta um caso de mapeamento de um *workflow* descrito num arquivo XML, produzido a partir do SGWfC Taverna, para a ferramenta GExpLine. Nesse exemplo, as atividades são representadas pelos elementos XML com nome *s:processor*, já os relacionamentos são determinados pelo nome de elementos XML *s:link* e as portas necessárias para a definição dos relacionamentos são os atributos dos elementos *s:link* com nome *source* e *sink*. Os atributos do tipo *source* dizem respeito a portas de origem dos relacionamentos (portas de saída das atividades), enquanto que os atributos do tipo *sink* referem-se às portas de destino (portas de entrada das atividades).

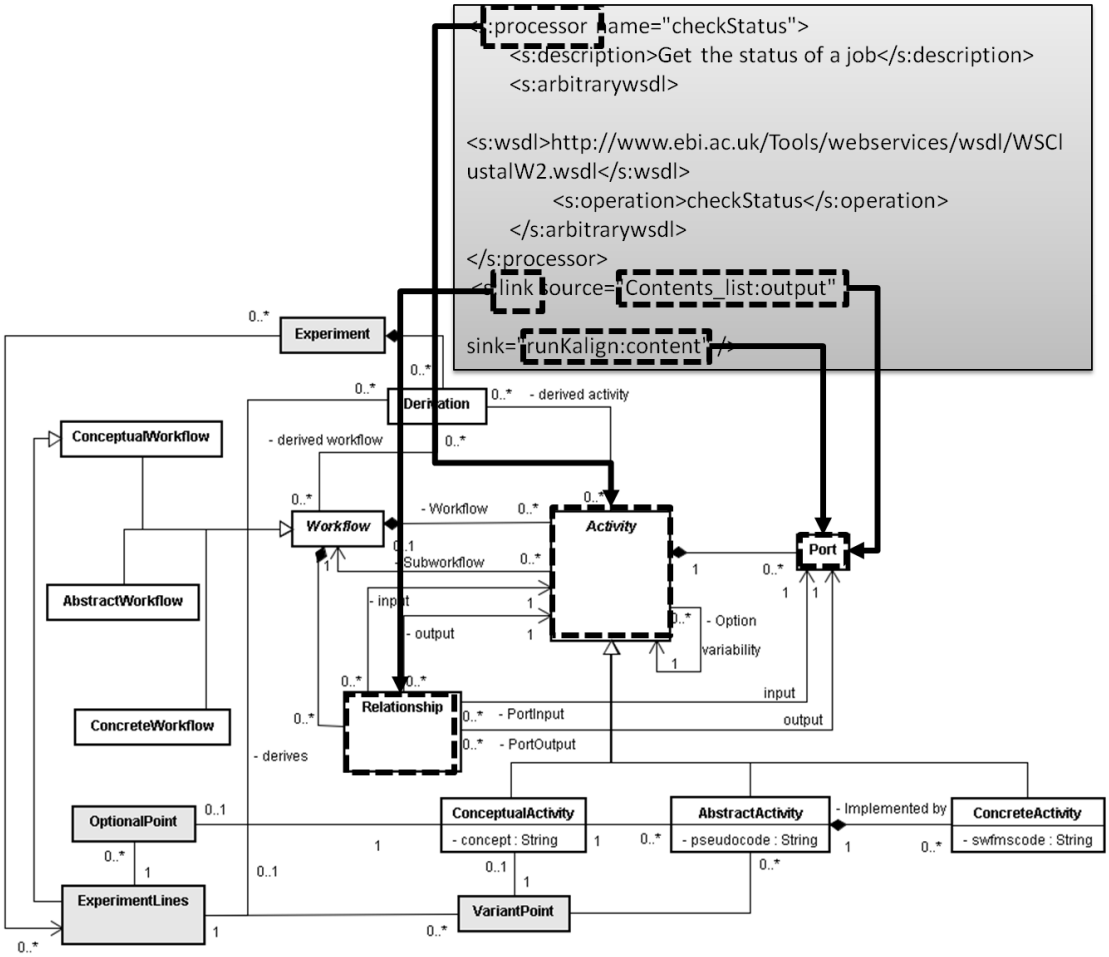

Figura 13. Mapeamento da estrutura dos *workflows* no esquema da GExpLine.

Para realizar a avaliação dos algoritmos de comparação e agrupamento dos *workflows* através do cálculo de similaridade, é necessário que uma quantidade expressiva de *workflows* seja utilizada. Com esse intuito, um conjunto de *workflows* foi importado do sítio *myExperiment* (GOBLE E ROURE, 2007). A importação baseou-se em arquivos no formato XML disponibilizados pelo repositório, que foram mapeados de acordo com o esquema de base relacional da ferramenta GExpLine, considerando-se apenas os *workflows* desenvolvidos no SGWfC Taverna. Em dezembro de 2010, o *myExperiment* apresentava 756 *workflows* do SGWfC Taverna, sendo destes 463 *workflows* públicos.

5.3. Cartucho de Comparação por Similaridade

O cartucho de comparação por similaridade tem o objetivo de realizar o cálculo de similaridade entre os *workflows*. Para isso, considera-se um conjunto de *workflows* concretos iniciais, que são analisados em pares de *workflows* e a similaridade entre os mesmos é calculada. A arquitetura SimiFlow permite também o desenvolvimento de diferentes cartuchos de comparação, contudo, nesse projeto, apenas uma implementação inicial é apresentada.

O algoritmo de comparação proposto para esse cartucho é dividido em duas fases. A primeira consiste na fase de penalidade, que realiza comparações estruturais dos elementos existentes nos *workflows*. O algoritmo considera inicialmente que os pares de *workflows* são iguais, ou seja, apresentam a mesma estrutura. A partir de diferentes critérios de comparação, a observação de diferenças entre os *workflows* implica em penalidades sobre o cálculo de similaridade. Já a segunda etapa realiza uma análise de todas as penalidades de cada par de *workflows* da fase anterior e calcula o valor de

similaridade a ser considerado para esses pares. Essa segunda parte é conhecida como fase de similaridade.

5.3.1. Fase de Penalidade

A fase de penalidade é a primeira fase do algoritmo de comparação. Como apresentado no Subcapítulo 5.3, essa fase realiza a comparação de pares de *workflows* a partir de um conjunto inicial de *workflows*, considerando todos os elementos existentes nesses *workflows*. Supondo a comparação das atividades de dois *workflows* W_1 e W_2. Para cada atividade de W_1 comparam-se cada atividade de W_2, tendo como parâmetro de comparação um conjunto de características das atividades, que são conhecidas pelo algoritmo como elementos. De uma maneira mais formal, o algoritmo efetua diferentes comparações de tuplas do tipo (A_i, A_j), onde A_i é uma atividade do *workflow* W_1 e A_j é uma atividade do *workflow* W_2.

Os elementos considerados pela comparação entre *workflows* são os seguintes: nome da atividade, tipo de atividade, porta, relacionamento e estrutura interna. A estrutura interna diz respeito à existência de *sub-workflow* ou a descrição da atividade concreta através de um conteúdo no formato XML. A partir dessa análise comparativa, valores de penalidades são atribuídos para cada elemento considerado. A nomenclatura adotada para cada elemento é apresentada na segunda coluna da Tabela 1.

Tabela 1. Variáveis de penalidade e valores máximos de penalidade.

Elemento	Variável	Penalidade Máxima
Nome da Atividade	PnA	512
Tipo da Atividade	PnT	1024
Porta	PnP	128
Relacionamento	PnR	64
Estrutura Interna (definição XML ou *sub-workflow*)	PnEI	256
Sub-workflow	PnSw	256
Definição XML	PnX	256

Pela Tabela 1, vale ressaltar também que as variáveis PnSw e PnX fazem referência a PnEI. Nesse caso, a penalidade de PnEI só pode ser igual à PnSw ou PnX, conforme apresentado abaixo:

$$\mathrm{PnEI}_{A_i,A_j} \begin{cases} \mathrm{PnSw}_{A_i,A_j}, \text{se } A_i \text{ ou } A_j \text{ é um sub}workflow \\ \mathrm{PnX}_{A_i,A_j}, \text{se } A_i \text{ e } A_j \text{ não são sub}workflows \end{cases}$$

Para cada par de atividades (A_i , A_j) comparadas, obtém-se uma penalidade total, sendo definida da seguinte forma:

$$\mathrm{PnTotal}_{A_i,A_j} = \mathrm{PnA}_{A_i,A_j} + \mathrm{PnT}_{A_i,A_j} + \mathrm{PnEI}_{A_i,A_j} + \mathrm{PnP}_{A_i,A_j} + \mathrm{PnR}_{A_i,A_j}$$

Além disso, os elementos considerados na comparação apresentarão valores máximos de penalidade, que são responsáveis por evidenciar a máxima interferência de um elemento na identificação de que dois *workflows* não são tão semelhantes. Esses

valores máximos de penalidade utilizados para esse algoritmo são apresentados na terceira coluna da Tabela 1.

O elemento mais simples de análise pelo algoritmo de comparação é o nome da atividade, que consiste na análise dos nomes de duas atividades, *A* e *A'*, por exemplo. Nesse caso, se for constatado que os nomes entre essas duas atividades são diferentes, a penalidade máxima para esse elemento é aplicada, ou seja, $\mathrm{PnA}_{A,A'} = 512$. Vale ressaltar também que caso os nomes das atividades sejam iguais, o valor da penalidade será nulo, portanto, $\mathrm{PnA}_{A,A'} = 0$. A Figura 14 apresenta um exemplo da comparação do nome das atividades.

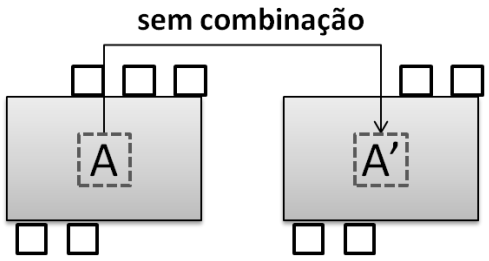

Figura 14. Comparação do nome da atividade.

O segundo elemento considerado é o tipo de atividade. Esse elemento considera o tipo específico de cada atividade de um *workflow* científico, independentemente do SGWfC. Existem diferentes tipos de atividades, como atividades de linha de comando e atividades de serviços *Web*, e as mesmas são analisadas da mesma forma que o nome de uma atividade. Portanto, considerando duas atividades *A* e *A'*, caso as duas atividades apresentem tipos diferentes, a penalidade máxima é aplicada, ou seja, $\mathrm{PnT}_{A,A'} = 1024$. A Figura 15 apresenta um exemplo para o tipo de atividade com penalidade máxima.

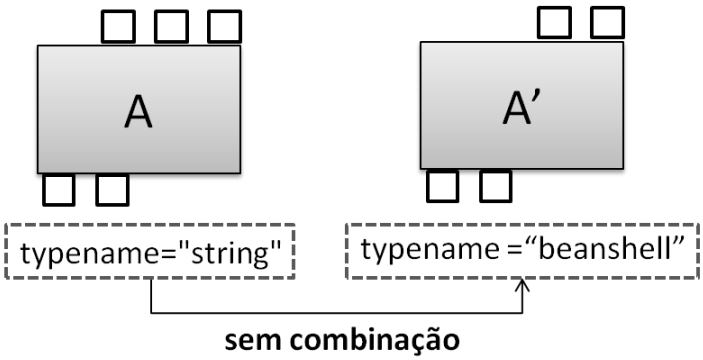

Figura 15. Comparação do tipo de atividade.

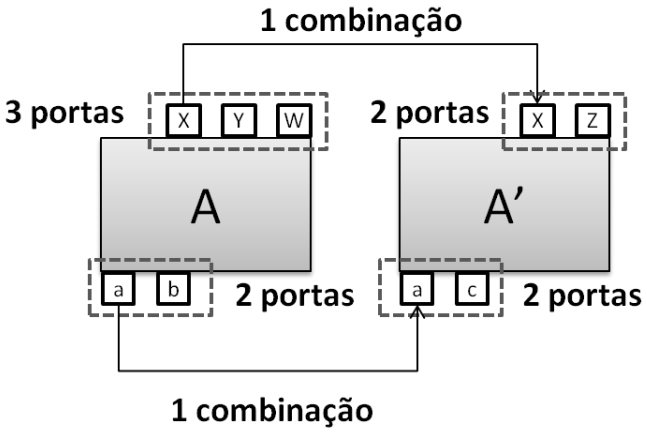

Figura 16. Comparação de conjunto de portas.

Além disso, o conjunto de portas da atividade é outro parâmetro considerado pelo algoritmo de comparação. Esse elemento apresenta como diferença em relação aos outros apresentados o fato dos valores de penalidades serem proporcionais ao número de portas identificadas como diferentes entre as atividades. Para realizar essa comparação, três tipos de portas são considerados: portas de entrada (PE), portas de saída (PS) e multiportas ou portas de entrada e saída (MP). A explicação desse elemento será baseada num exemplo teórico, em que há duas atividades *A* e *A’*, sendo que a atividade *A* possui três portas de entrada (*X*, *Y* e *W*) e duas portas de saída (*a* e *b*),

enquanto a atividade *A'* apresenta apenas duas portas de entrada (*X* e *Z*) e duas portas de saída (*a* e *c*). A Figura 16 representa esse exemplo teórico.

A primeira etapa desse elemento considera a comparação de todas as portas de entrada da atividade *A* com as portas de entrada da atividade *A',* avaliando se o nome e o tipo dessas portas são iguais. No final dessa etapa, o número de portas iguais é contabilizado. No exemplo teórico, pode-se observar que existe apenas uma combinação (porta *X*), ou seja, uma porta da atividade *A* que é igual à outra porta da atividade *A'*. Como a penalidade é proporcional ao número de portas diferentes, realiza-se o cálculo de um fator de proporcionalidade α_e para as portas de entrada. A fórmula abaixo apresenta o cálculo do fator α_e, sendo que PE de A representa o número de portas de entrada da atividade *A* e PE de A' representa o número de portas de entrada da atividade *A'*.

$$\alpha_{\mathrm{e}} = \frac{(\mathrm{PE\ de\ A} + \mathrm{PE\ de\ A'}) - (2 \times \mathrm{combinações})}{(\mathrm{PE\ de\ A} + \mathrm{PE\ de\ A'})} = \frac{(3+2)-(2\times 1)}{(3+2)} = \frac{3}{5}$$

O mesmo cálculo apresentado na fórmula acima é realizado pelas portas de saída e pelas multiportas, sendo os fatores de proporcionalidade expressos por α_s e α_m, respectivamente.

$$\alpha_{\mathrm{s}} = \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}$$

$$\alpha_{\mathrm{m}} = \frac{(0+0)-(2\times 0)}{(0+0)} = 0$$

Para calcular o valor final de penalidade do conjunto de portas de uma atividade com outra, calcula-se a média dos fatores de proporcionalidade α_e, α_s e α_m e, depois,

multiplica-se esse resultado pela penalidade máxima para a comparação de portas, portanto:

$$\mathrm{PnP}_{\mathrm{A,A'}} = \left(\frac{\alpha_e + \alpha_s + \alpha_m}{3} \times 128\right) = \left(\frac{\frac{3}{5} + \frac{1}{2} + 0}{3}\right) \times 128 \cong 46{,}93$$

Além das propriedades relacionadas às atividades e portas, o algoritmo de comparação analisa um conjunto de relacionamentos de entrada (RE) e outro de saída (RS) para uma atividade. A penalidade aplicada aos relacionamentos também apresenta uma proporcionalidade em relação ao número de diferenças identificadas entre os pares de *workflows*. É importante enfatizar que se consideram os relacionamentos de entrada e de saída. Portanto, a similaridade entre os relacionamentos está relacionada à que atividades eles estão conectados. Para facilitar o entendimento, a explicação desse algoritmo será baseada num exemplo teórico.

O exemplo considera a comparação entre duas atividades, *A* e *A'*. A atividade *A* possui dois relacionamentos de saída, que conectam as atividades *A* e *B*; e *A* e *C*. Ao mesmo tempo, a atividade *A'* apresenta dois relacionamentos de saída, que conectam as atividades *A'* e *B'*; e *A'* e *C*. A Figura 17 é uma representação desse exemplo. Para a determinação da similaridade dos relacionamentos entre duas atividades, avalia-se se os relacionamentos estão conectados a atividades com o mesmo nome. Dessa forma, como as atividades dos relacionamentos de saída são iguais (atividade *C*), o relacionamento entre as atividades *A* e *C* é similar ao relacionamento entre as atividades *A'* e *C*.

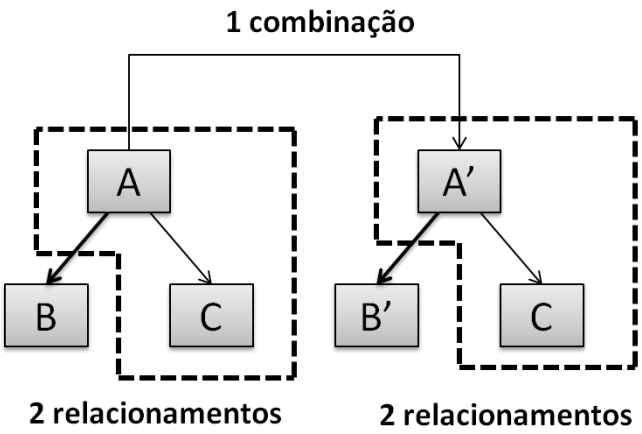

Figura 17. Comparação de relacionamentos.

Assim como no caso das portas, os relacionamentos apresentam seu cálculo baseados em fatores de proporcionalidade. Dessa vez, contudo, os fatores estão associados aos relacionamentos de entrada e de saída, sendo representados por α_e e α_s, respectivamente. Como não existem relacionamentos de entrada para a comparação entre as atividades *A* e *A'*, então $\alpha_e = 0$, enquanto que α_s pode ser calculado da seguinte forma:

$$\alpha_s = \frac{(\text{RS de A} + \text{RS de A}') - (2 \times \text{combinações})}{(\text{RS de A} + \text{RS de A}')} = \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}$$

Já para o cálculo final da penalidade dos relacionamentos sobre um par de atividades, nesse caso, atividades *A* e *A'*, deve-se realizar a média dos valores de α_e e α_s e multiplicar pelo valor de penalidade máxima para os relacionamentos. Logo:

$$\text{PnR}_{\text{A,A}'} = \left(\frac{\alpha_e + \alpha_s}{2} \times 64\right) = \left(\frac{0+\frac{1}{2}}{2} \times 64\right) = 16$$

O último item utilizado para análise e penalização é a estrutura interna das atividades. A estrutura interna pode ser de dois tipos: definição XML e *sub-workflows*. O primeiro tipo diz respeito a atividades concretas que apresentam a sua descrição através de elementos XML, enquanto que o segundo refere-se à existência de um *workflow* na definição das atividades (ocorrência de *sub-workflows*). Sendo assim, para realizar a comparação desse item entre as atividades, há três possibilidades de situações: as atividades apresentam suas definições no formato XML; as atividades apresentam um *workflow* definido dentro delas (ocorrência de *sub-workflows*) ou; uma atividade apresenta sua definição no formato XML e a outra atividade apresenta um *workflow* definido dentro dela.

No caso de duas atividades apresentarem sua definição no formato XML (ou seja, as atividades não apresentam *sub-workflows*), consideram-se como elementos de comparação as definições em XML (*tags*) de cada atividade. Portanto, dois elementos são considerados similares se apresentam os mesmos valores das *tags* existentes no XML. Apesar dessa avaliação objetiva (em verdadeiro ou falso), o resultado da penalidade assemelha-se ao caso das portas e dos relacionamentos, apresentando um valor proporcional ao número de elementos diferentes existentes no XML entre duas atividades.

A Figura 18 apresenta um exemplo de duas atividades com suas definições no formato XML (atividades *A* e *A'*). Nesse caso, as *tags* denominadas "s:processor" não são consideradas no cálculo da penalidade, pois esse valores referem-se a definição de uma atividade no SGWfC Taverna, que foi utilizado para realizar a importação dos *workflows*. Além desses, em cada atividade há dois elementos a serem utilizados na comparação, o elemento com *tag* "s:description" e outro com a *tag* "s:arbitrarywsdl". Além disso, a figura enfatiza através das chaves que trecho dos elementos é considerado

na comparação. Pelo conteúdo das definições no formato XML, pode-se perceber que apenas um caso (*tag* igual a "s:description"), os valores serão iguais, representando um caso de combinação.

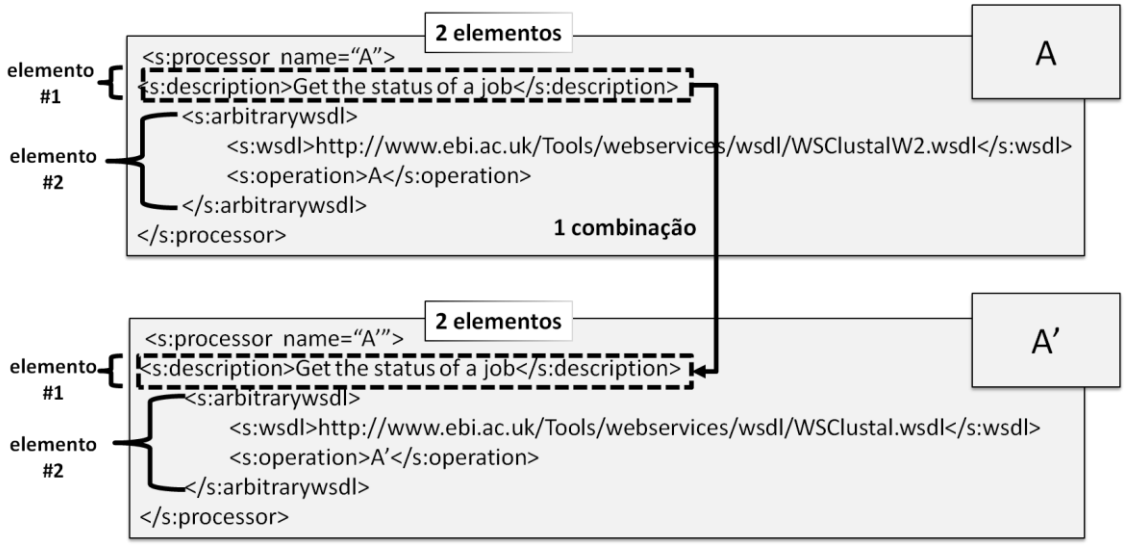

Figura 18. Comparação de atividades com definições no formato XML.

A partir da Figura 18, observa-se que o fator de proporcionalidade α pode ser calculado da seguinte maneira:

$$\alpha = \frac{(\text{Elem. de A} + \text{Elem. de A}\prime) - (2 \text{ combinações})}{(\text{Elem. de A} + \text{Elem. de A}\prime)} = \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}$$

Dessa forma, o fator de proporcionalidade pode ser multiplicado pela penalidade máxima, obtendo o valor final da penalidade sobre a definição de atividades no formato XML:

$$\text{PnX}_{\text{A,A}\prime} = (\alpha \times 256) = \left(\frac{1}{2} \times 256\right) = 128$$

Considerando-se o caso em que as duas atividades são *sub-workflows*, o algoritmo realiza uma chamada recursiva de cada *sub-workflow*, executando as fases de penalidade e de similaridade novamente. O resultado desse processo recursivo será um fator de dissimilaridade λ, cujo valor pode variar de 0 a 1. A multiplicação desse fator pela penalidade máxima obtém a penalidade final para esse caso.

A Figura 19 representa um caso teórico de duas atividades que são *sub-workflows*. A atividade *A* é composta de outras três atividades, sendo elas *A1*, *A2* e *A3*. Enquanto isso, a atividade *A'* também é composta de três atividades, cujos nomes são *A'1*, *A'2* e *A'3*. Sendo essa a configuração das atividades, pode-se concluir que o processo recursivo de invocação do algoritmo de comparação será realizado. Supondo que o fator de dissimilaridade λ, calculado por esse exemplo seja $\lambda = 0{,}55$, então o valor final da penalidade do *sub-workflow* é igual a $\mathrm{PnSw}_{A,A'} = 256 \times \lambda = 256 \times 0{,}55 = 140{,}8$.

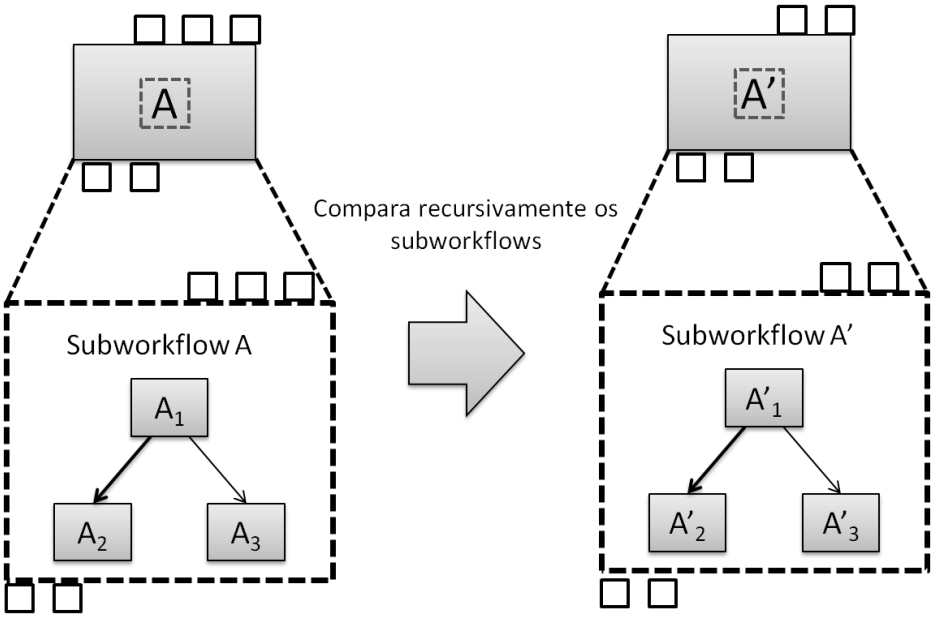

Figura 19. Comparação de atividades que são *sub-workflows*.

Por último, o caso em que se considera uma atividade com definição no formato XML e outra que é um *sub-workflow*, o valor de penalidade final para a comparação é igual à penalidade máxima, ou seja, $\mathrm{PnSw}_{A,A'} = 256$. A penalidade é considerada máxima, nessa situação, pois as duas estruturas não oferecem nenhum mecanismo possível de comparação, sendo ditas diferentes quanto à própria estrutura ou representação. Um exemplo desse caso é apresentado na Figura 20.

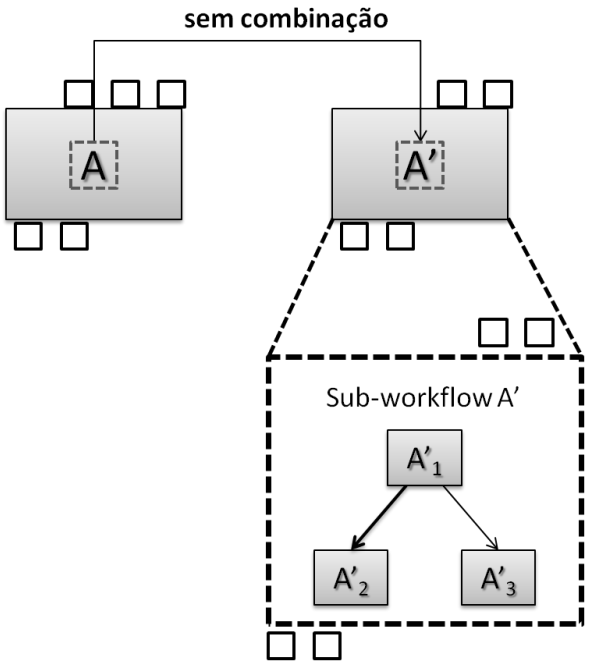

Figura 20. Comparação de uma atividade com definição XML e outra que é um *sub-workflow*.

5.3.2. Fase de Similaridade

Após a execução da fase de penalidade, a fase de similaridade é inicializada pelo algoritmo. Nessa fase, a similaridade total entre os *workflows* W_x e W_y é calculada através de um algoritmo guloso (CORMEN *et al*., 2009), dado que todos os cálculos de

penalidade total entre pares de atividades (A_i, A_j) foram efetuados na etapa de penalidade. No primeiro momento, o algoritmo realiza uma ordenação crescente da penalidade total ($PnTotal$) de pares de atividades com o intuito de obter os pares de atividades que sejam muito similares, pois quanto menor a penalidade sobre uma comparação de atividades, menor é o número de diferenças entre as atividades (necessário considerar o peso de cada elemento analisado – penalidade máxima).

Além disso, o algoritmo considera também a possibilidade de uma atividade de um *workflow* não ser semelhante a nenhuma atividade de outro *workflow*. Nesse caso, os pares formados são de cada atividade com outra atividade vazia, apresentando penalidade igual ao valor da penalidade máxima (soma de todas as penalidades máximas), sendo este igual a 1.984. Assim, esse resultado é adicionado ao final da lista ordenada de penalidades. Um exemplo teórico pode ser dado pela comparação de uma atividade A_i do *workflow* W_x com uma atividade A_j do *workflow* W_y, sendo que as atividades A_i e A_j não são similares. Portanto, a atividade A_i apresenta um par (A_i, Ø), sendo que Ø denota uma atividade vazia. O mesmo vale para a atividade A_j, cujo par é (A_j, Ø).

Após a ordenação da lista de PnTotal, o algoritmo inicializa o processo de similaridade obtendo o primeiro par (A_i, A_j) da lista crescente, ou seja, seleciona-se o par com menor penalidade. Esse par com menor valor de PnTotal é conhecido por $\mathrm{PnTotal}_1$. Ao mesmo tempo, esse par é considerado o par com maior similaridade entre todos os pares da lista ordenada. O fato de não realizar uma análise com outros pares dessa lista enfatiza a característica de um algoritmo guloso. Portanto, todos os pares da lista que apresentam essas atividades mais similares são descartados dos pares a serem analisados futuramente. E assim continua a execução do algoritmo, determinando os próximos pares, ao mesmo tempo em que armazena os valores de PnTotal e a remoção

dos pares que não são mais pertinentes. O ponto de parada do algoritmo é identificado quando a lista ordenada está vazia.

Ao final desse processo, o algoritmo guloso terá obtido *N* pares de atividades, sendo que esses pares possuem os menores valores possíveis de $PnTotal$ entre duas atividades. A partir dessas penalidades armazenadas e do número de pares obtidos, o fator de dissimilaridade (λ), que identifica o grau de diferenciação entre dois *workflows* pode ser calculado da seguinte forma:

$$\lambda = \frac{\mathrm{PnTotal}_1 + \mathrm{PnTotal}_2 + \ldots + \mathrm{PnTtotal}_N}{1984 \times N}$$

Sendo assim, o fator de similaridade (λ′) pode ser calculado como abaixo:

$$\lambda' = 1 - \lambda$$

5.4. Cartucho de Agrupamento

Após a execução do algoritmo de comparação, presente no cartucho de comparação por similaridade, há a separação dos *workflows* existentes em grupos de *workflows* semelhantes. Para isso, o cartucho de agrupamento obtém todos os valores de similaridade entre pares de *workflow* existentes no esquema da base de dados da ferramenta GExpLine e define um grafo completo *G* (CORMEN *et al*., 2009). Um grafo pode ser definido como completo, quando cada vértice do grafo apresenta uma aresta com todos os outros vértices. O grafo completo *G* possui *n* vértices, sendo que os vértices são representados por todos os *workflows* existentes no esquema da GExpLine

e *m* arestas, que referem-se ao valor de similaridade entre dois *workflows*. Portanto, o valor de *m* é definido de acordo com a combinação C_2^n.

Após a criação desse grafo completo, a SimiFlow realiza a remoção das arestas que apresentam um valor de similaridade menor que o valor de corte. A partir da execução desse corte, uma floresta de grafos é formada, apresentando grafos desconexos (CORMEN *et al*., 2009). Além disso, os grafos desconexos que só apresentam um vértice são removidos, mantendo apenas os grafos desconexos com mais de um vértice, sendo esses remanescentes, os grupos de *workflows* científicos formados.

O valor de corte, ou valor limite para corte, é uma entrada de parâmetros existente no algoritmo nesse cartucho de agrupamento. Esse valor determina quanto deve ser o valor de similaridade para que dois *workflows* sejam considerados pertencentes ao mesmo grupo. A escolha desse valor é uma decisão *ad-hoc*, sendo necessário que haja um conhecimento ou experiência para a sua determinação. Portanto, para um determinado domínio, a escolha de um valor de corte igual a 0,70 pode ser suficiente para afirmar que dois *workflows* são do mesmo grupo, enquanto que para outra aplicação, esse valor pode não ser conveniente. Essa situação ocorre em outros algoritmos de agrupamento, em que há a necessidade de determinar um parâmetro de entrada, como acontece com o K-means, sendo K o número de clusters para o parâmetro de entrada. Nessa situação, a escolha de um valor inapropriado para K pode acarretar num resultado ruim no processo de agrupamento (CORMEN *et al*., 2009).

A Figura 21 apresenta um exemplo de agrupamento de *workflows*. Nesse exemplo, existem quatro *workflows* sendo que, quando aplicado um valor de corte igual a 0,55, dois grupos de *workflows* podem ser formados, pois todas as arestas com valor de similaridade inferior a 0,55 serão removidas. Sendo assim, os *workflows* #1 e #2 pertencem ao mesmo grupo, enquanto que os *workflows* #3 e #4 pertencem a um

segundo grupo. Vale ressaltar também que outros algoritmos de agrupamento podem ser testados na SimiFlow, uma vez que essa arquitetura foi projetada de forma expansível.

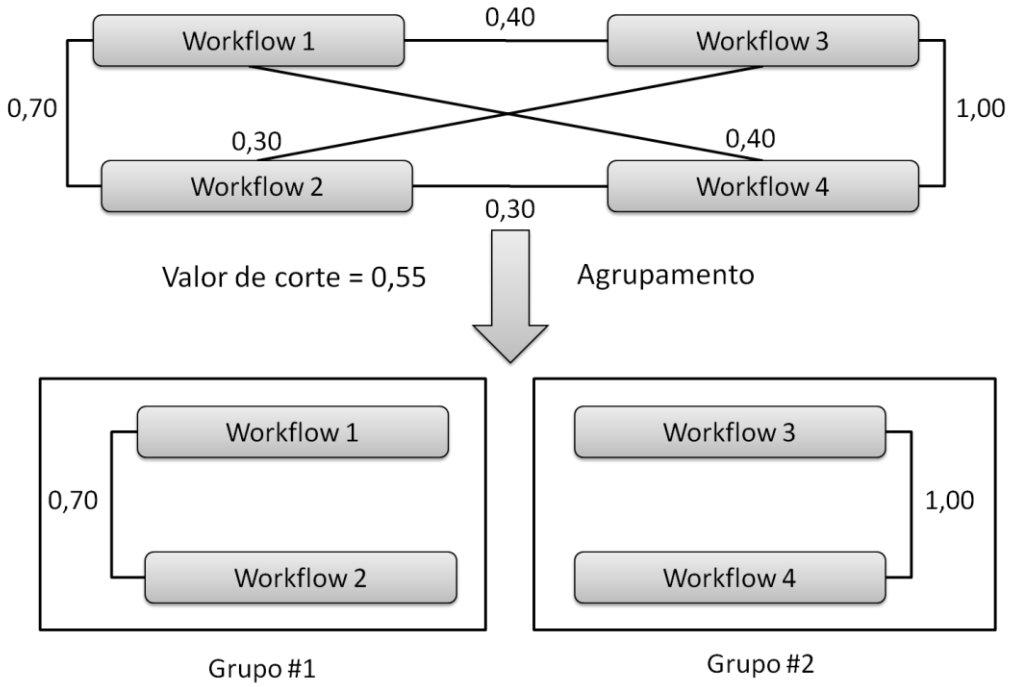

Figura 21. Exemplo de agrupamento de *workflows*.

5.5. Trabalhos Relacionados

Um dos algoritmos existentes para agrupamento de *workflows* é proposto por (SANTOS *et al*., 2008), sendo que esse apresentou-se como o mais semelhante à arquitetura SimiFlow. Nessa abordagem, os *workflows* são representados por grafos, através de métricas de subgrafos (BUNKE E SHEARER, 1998), ou por *arrays* multidimensionais, utilizando a medida de distância do cosseno para definir o quanto dois elementos são semelhantes. Para realizar o agrupamento, os algoritmos K-means (HAN E KAMBER, 2006), K-medoids (HAN E KAMBER, 2006) e hierárquico (JAIN

et al., 1999) foram utilizados. Em função da representação do *workflow* usando grafos ou *arrays*, as propriedades específicas de *workflows* científicos não foram oferecidas, como, por exemplo, a definição de portas de entrada.

Uma metodologia também semelhante à SimiFlow é proposta por (JUNG E BAE, 2006), tendo como foco o apoio a análise de repositórios de processos. Nesse trabalho, os *workflows* são agrupados de acordo com o cálculo de similaridade entre as atividades de um *workflow* a partir da medida do cosseno. Algumas propriedades específicas dos *workflows* não são oferecidas, uma vez que esse algoritmo utiliza sua representação por *arrays* multidimensionais.

Outros mecanismos e técnicas de comparação existem para analisar modelos de dados. O trabalho de (OHST *et al*., 2003) utiliza um método baseado na comparação de versões diferentes de diagramas UML, através da detecção de diferenças em sistemas de gerência de configuração (SCM, do inglês *Software Configuration Management*). Nesse trabalho são consideradas duas categorias de modificação de documentos UML, que são representados por grafos acíclicos na forma de uma árvore geradora (CORMEN *et al*., 2009). Essas categorias de documentos são baseadas na detecção das diferenças interna dos nós, enquanto que a outra realiza uma análise estrutural do documento UML. Nesse caso, o algoritmo compara os diferentes níveis da árvore geradora com todos os outros. Além disso, ao contrário do que acontece com esse trabalho, a SimiFlow não representa os *workflows* através de um grafo.

Outra abordagem existente baseia-se na comparação de diagramas UML (UHRIG, 2008) com baixo custo computacional. O algoritmo compara os diagramas UML com todos os outros, utilizando como métrica a classe dos diagramas. Após essa etapa, um grafo bipartito é gerado, sendo que cada aresta representa o custo entre dois diagramas.

As limitações são semelhantes aos casos demonstrados anteriormente ao representar *workflows* por grafos.

O SiDiff (SiDiff, 2010) é outra proposta semelhante a SimiFlow, em que utiliza algoritmos de comparação e *merge* na análise dos modelos de dados. O algoritmo de comparação existente nessa abordagem é semelhante ao de (OHST *et al*., 2003). Da mesma forma que acontece com a SimiFlow, o SiDiff realiza o cálculo de similaridade a partir de penalidades entre os elementos diferentes.

O sítio do *myExperiment* também propõe o agrupamento de *workflows* científicos (GODERIS *et al*., 2008), através da organização dos *workflows* em grupos com o intuito de serem reutilizados em outros experimentos. Além disso, a comparação é realizada pela descrição dos *workflows*, ao invés de serem avaliados quanto às suas estruturas. A principal vantagem da SimiFlow em relação a esse mecanismo é a independência do SGWfC, pois a proposta do *myExperiment* apenas suporta *workflows* no formato do Taverna.

6. Avaliação Experimental

Esse capítulo tem o propósito de apresentar a avaliação experimental realizada na arquitetura SimiFlow. Essa avaliação experimental foi dividida, por sua vez, em dois estudos. O primeiro é responsável pela comparação e análise de um conjunto reduzido de *workflows* científicos, a fim de avaliar se os algoritmos propostos para a comparação e o agrupamento obtiveram resultados coerentes com os *workflows* existentes. Posteriormente, o segundo estudo realiza o mesmo processo considerando todos os *workflows* importados a partir do sítio *myExperiment*, visando avaliar se a técnica de comparação é computacionalmente tratável. Dessa forma, esse capítulo foi dividido da seguinte maneira: o Subcapítulo 6.1 apresenta a preparação dos dados para os estudos realizados; o Subcapítulo 6.2 demonstra as configurações necessárias para a execução dos experimentos; o Subcapítulo 6.3 apresenta o estudo focado na análise dos algoritmos de comparação por similaridade e de agrupamento de *workflows*; e, por último, o Subcapítulo 6.4 apresenta o estudo realizado sobre os *workflows* do repositório *myExperiment*.

6.1. Preparação dos dados

A preparação dos dados consistiu na importação dos *workflows* existentes no sítio *myExperiment*. Apenas *workflows* do SGWfC Taverna foram considerados. Os dados desse processo são referentes a dezembro de 2010.

Sendo assim, o sítio *myExperiment* proporcionou o armazenamento de 463 *workflows* científicos no esquema da GExpLine, sendo que foi necessária 1 hora e 40

minutos para a realização da carga. Portanto, o processo de comparação por similaridade considerando-se a base completa necessitara efetuar 106.953 comparações (C_2^{463}) entre pares de *workflows* (SILVA *et al*., 2011).

6.2. Configuração do Ambiente

A etapa de execução dos experimentos tem o propósito de apresentar o mecanismo de paralelização adotado para o processamento dos algoritmos de comparação por similaridade e de agrupamento de *workflows*. A necessidade de paralelização para o algoritmo de comparação é justificado pela existência de uma elevada quantidade de comparações a serem efetuadas, necessitando de muito tempo, caso esse processo seja sequencial.

Logo, para realizar as comparações, utilizou-se uma máquina SGI Altix ICE 8200 com 64 nós (Intel Xeon 8-core processor) e o Chiron (OGASAWARA *et al*., 2011), uma máquina de execução de *workflow* científico inspirada no modelo relacional, que foi utilizada para a paralelização das comparações. Esse subcapítulo apresenta a configuração para a execução do primeiro estudo, que utilizou quatro nós de 8 processadores (32 cores).

Como o Chiron é baseado no modelo relacional, suas atividades são caracterizadas por realizar operações algébricas adaptadas ao contexto de *workflows* científicos. Assim, os dados dos *workflows* são representados por relações, sendo que cada relação possui diversas tuplas. No caso da arquitetura SimiFlow, o *workflow* modelado no Chiron apresenta duas atividades, sendo que a primeira realiza a comparação de cada par de *workflows*, enquanto a segunda é responsável pela execução do algoritmo de agrupamento.

O Chiron realiza a especificação do *workflow* através de um arquivo no formato XML. A Figura 22 apresenta o arquivo XML utilizado para configurar a execução da SimiFlow. Dentro do elemento principal *Chiron*, o elemento *database* especifica as características da base de dados a ser utilizada. Por conseguinte, o elemento *Workflow* define as atividades existentes por meio do elemento *Activity*. As atividade determinadas para a SimiFlow são identificadas pelas *tags* "comparison" e "prune", sendo estas representadas por operações algébricas do tipo *Map* (OGASAWARA *et al*., 2011).

Por ser uma aplicação MPI, o Chiron executa a SimiFlow em paralelo. O Chiron foi desenvolvido na linguagem de programação Java e utilizou um MPI específico para essa linguagem, denominado de MPJ (CARPENTER *et al*., 2000) (BAKER *et al*., 2008). Para esse primeiro estudo utilizou-se 4 nós de 8 processadores (32 cores) na máquina apresentada anteriormente. A submissão do job necessita do desenvolvimento de um script PBS (BAYUCAN *et al*., 2000), em que apresenta um número de nós e processadores utilizados (primeira linha do script) e a chamada do Chiron (linhas 16, 17 e 18). A Figura 23 apresenta o script PBS para a comparação de *workflows*.

```
<?xml version="1.0" encoding="UTF-8"?>
<Chiron>
   <database name="chiron-simiflow" server="146.164.31.200"
port="5432" username="postgres" password="postgres"/>
   <Workflow execmodel="DYN_FTF" tag="exp" description="exp"
exectag="simiflow" expdir="/home/users/silva/simiflow/exp">
      <Activity tag="comparison" type="map" activation="java -
jar /home/users/silva/bin/gexpline.jar compare %=WORK1%
%=WORK2%">
         <Relation reltype="Input" name="rel_in_1"
filename="input_1.txt"/>
         <Relation reltype="Output" name="rel_out_1"
filename="output_1.txt"/>
         <Field name="WORK1" type="float" input="rel_in_1"
output="rel_out_1" decimalplaces="0"/>
         <Field name="WORK2" type="float" input="rel_in_1"
output="rel_out_1" decimalplaces="0"/>
         <Field name="PRUNE" type="float" input="rel_in_1"
output="rel_out_1" decimalplaces="0"/>
      </Activity>
      <Activity tag="prune" type="map" activation="java -jar
/home/users/silva/bin/gexpline.jar prune %=PRUNE%">
         <Relation reltype="Input" name="rel_in_2"
filename="input_2.txt"/>
         <Relation reltype="Output" name="rel_out_2"
filename="output_2.txt"/>
         <Field name="WORK1" type="float" input="rel_in_2"
output="rel_out_2" decimalplaces="0"/>
         <Field name="WORK2" type="float" input="rel_in_2"
output="rel_out_2" decimalplaces="0"/>
         <Field name="PRUNE" type="float" input="rel_in_1"
output="rel_out_1" decimalplaces="0"/>
      </Activity>
   </Workflow>
</Chiron>
```

Figura 22. Especificação do *workflow* para a execução da SimiFlow pelo Chiron

```
#PBS -l nodes=4:ppn=8
#PBS -l walltime=40:00:00
#PBS -S /bin/bash
#PBS -N simiflow-1
#PBS -j oe
#PBS -V
#PBS -m ae
#PBS -M oi
# change directory
cd ${PBS_O_WORKDIR}
# create list with names from data network
echo Java Job: ${PBS_JOBID}
sort ${PBS_NODEFILE} | uniq > ~/tmp/${PBS_JOBID}
which java
cd $MPJ_HOME/bin
java -jar mpjEnv.jar ~/tmp/${PBS_JOBID}
./mpjmanual.sh ~/tmp/${PBS_JOBID}.conf $HYDRA_V2/chiron.jar MPI 8
/home/users/silva/simiflow/1/chiron.xml
```

Figura 23. Script PBS para submissão de job

Além disso, há a necessidade de identificar o número de processos que irão se comunicar durante a execução da aplicação, assim como suas respectivas portas e identificadores. Para isso, um arquivo de configuração foi desenvolvido para especificar essas propriedades para o MPJ. A Figura 24 demonstra como foi desenvolvido esse arquivo para o caso da arquitetura SimiFlow para 32 cores.

```
# Number of Processes
4
# Protocol switch limit
131072
# Entry in the form of machinename@port@rank
r1i0n1@20000@0
r1i0n2@20002@1
r1i0n3@20004@2
r1i0n4@20006@3
```

Figura 24. Arquivo de configuração do MPJ.

6.3. Estudo dos algoritmos

O primeiro estudo da arquitetura SimiFlow foi realizado com o intuito de analisar o comportamento dos algoritmos de comparação por similaridade e de agrupamento de *workflows*. Neste primeiro experimento foi feito um recorte da base importada no *myExperiment*, selecionando-se apenas os *workflows* que começam com o nome "EBI". Essa decisão baseou-se na demanda de muito tempo para executar as comparações de todos os *workflows* existentes na base de dados da ferramenta GExpLine. O resultado dessa seleção foi a diminuição do número total de *workflows* para 36 *workflows*, ou seja, 630 comparações (C_2^{36}).

Após a execução dos algoritmos, diferentes grupos de *workflows* são formados, tendo como parâmetro de entrada os pares de *workflows* a serem comparados (para a comparação) e o valor de corte (para o agrupamento de *workflows*). A Tabela 2 apresenta os grupos de *workflows* produzidos pelo processo de agrupamento com valor de limite de corte igual a 0,70. Já a Figura 25 apresenta o grafo formado, representando os grupos e as arestas que não foram cortadas.

Pelos grupos #1, #2, #3, #4, #5, #6 e #7, observou-se que os *workflows* apresentaram uma estrutura semelhante entre os seus componentes. A Figura 26 apresenta um exemplo de dois *workflows* semelhantes que foram alocados num mesmo grupo (#7), cujo valor de similaridade entre eles é de 0,81. Assim como acontece com esses exemplos, foi possível observar, para esse primeiro estudo, que os pares de *workflows* apresentaram estruturas semelhantes numa proporção equivalente ao valor de similaridade calculado. Sendo assim, pode-se constatar que os *workflows* concretos alocados em um mesmo grupo, poderiam dar origem a linhas de experimentos, dado os conceitos de variabilidade e opcionalidade.

Tabela 2. Agrupamento de *workflow*

Nome do *Workflow*	Número do Grupo
EBI InterProScan v1	**#1**
EBI InterProScan v2	**#1**
EBI_FASTA	**#2**
EBI_blastpgp_PSI-BLAST	**#2**
EBI_NCBI_BLAST	**#2**
EBI FASTA with prompts	**#3**
EBI_NCBI_BLAST_with_prompts	**#3**
EBI_ClustalW2_phylogentic_tree	**#4**
EBI_ClustalW2	**#4**
EBI_InterProScan_tmhmm_signalp	**#5**
EBI_InterProScan v3	**#5**
EBI_ScanPS	**#6**
EBI_Phobius	**#6**
EBI_MPsrch	**#6**
EBI_MaxSprout	**#7**
EBI_Kalign	**#7**
EBI_TCoffee	**#7**
EBI_MUSCLE	**#7**
EBI_MAFFT	**#7**

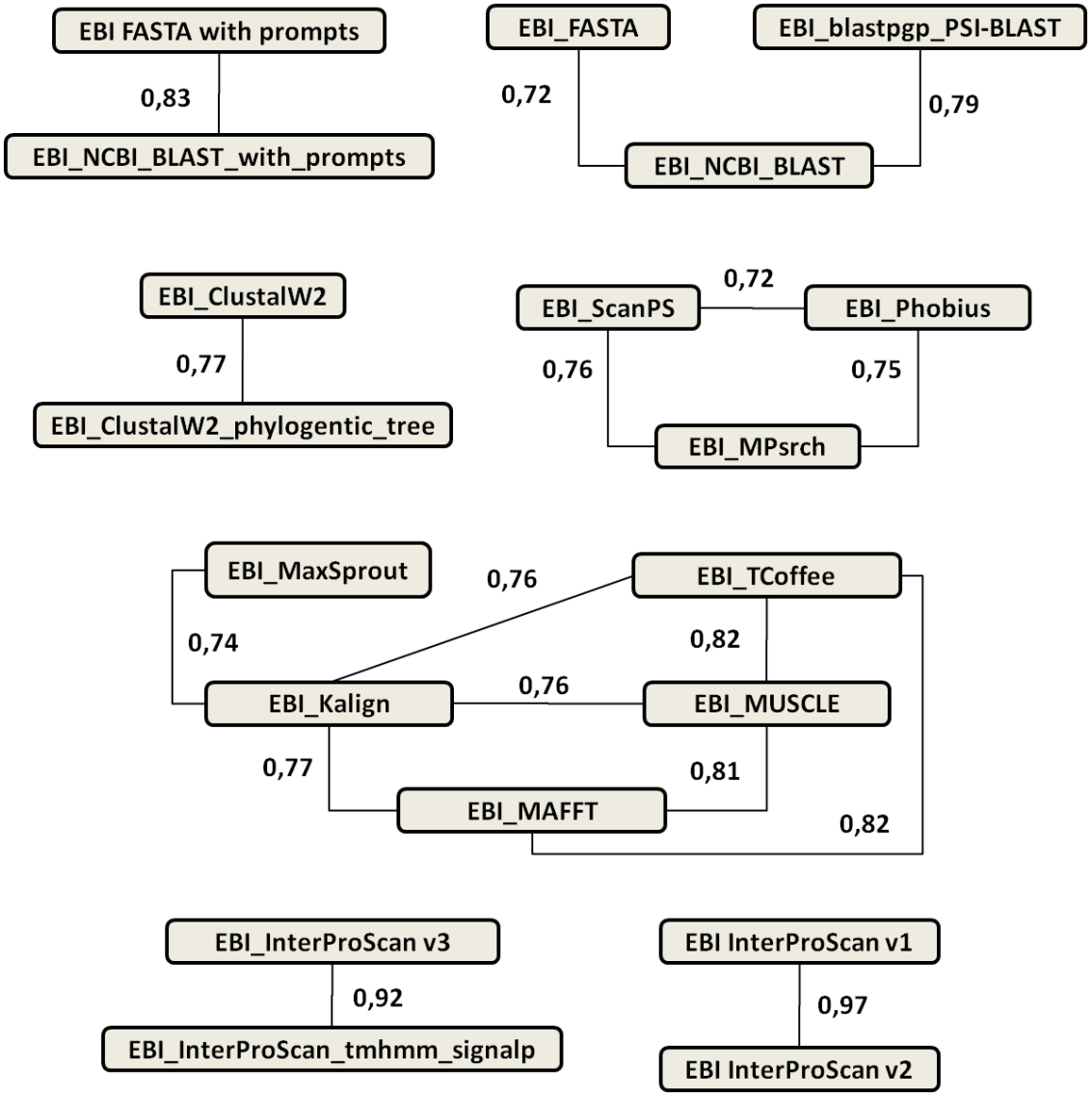

Figura 25. Representação do grafo formado após o agrupamento dos *workflows*.

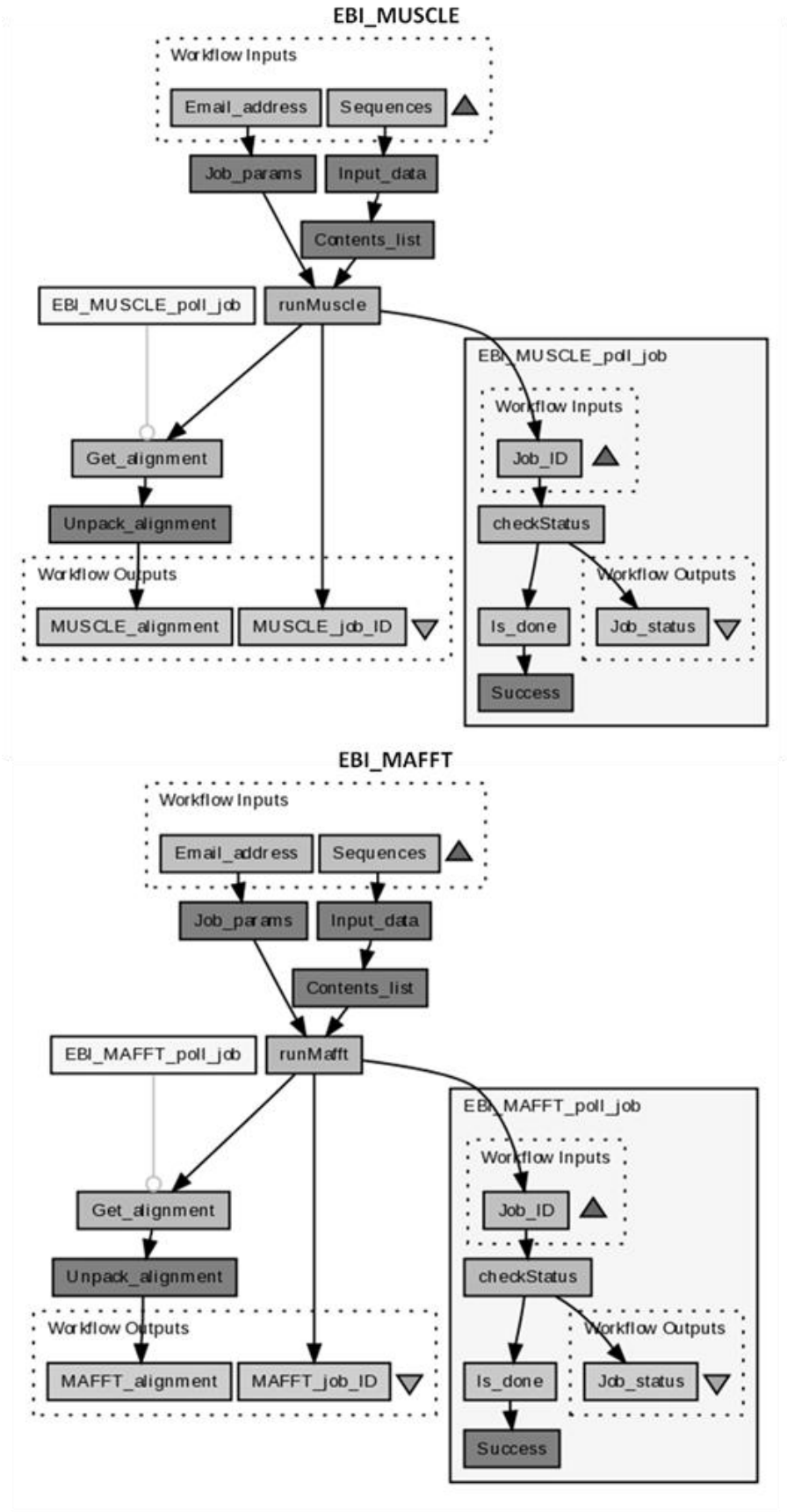

Figura 26. Exemplo de dois *workflows* que foram alocados num mesmo grupo.

6.4. Estudo do repositório *myExperiment*

O segundo estudo foi realizado com o intuito de analisar se a execução dos algoritmos (comparação e agrupamento de *workflows*) é computacionalmente tratável, considerando-se um número maior de *workflows*. Nesse caso, utilizou-se o repositório *myExperiment*. Portanto, todos os pares de *workflows* foram comparados e agrupados, considerando um valor de limite de corte igual a 0,70, da mesma forma que aconteceu para o primeiro estudo. Apesar da obtenção dos grupos e dos valores de similaridade, um estudo analítico completo não foi possível, uma vez que existem muitos *workflows* a serem observados quanto a sua estrutura. Vale ressaltar que a execução do segundo estudo utilizou as mesmas configurações do Chiron apresentadas no primeiro estudo, só que, nesse caso, considerando apenas 1 nó de 8 processadores (8 cores).

Para analisar a vantagem no uso do Chiron para paralelização de todas as comparações realizadas pela arquitetura SimiFlow, as métricas de *speedup* e de eficiência foram utilizadas para essa avaliação (HILL, 2006). A métrica de *speedup* (S_p) consiste na relação entre o tempo de execução sequencial do algoritmo (S_1) e o tempo de execução paralelo do algoritmo (T_p), considerando-se p processadores. Enquanto isso, a eficiência (E_p) é definida pela relação entre o valor de *speedup* e o número de processadores utilizados (p). Portanto:

$$S_p = \frac{T_1}{T_p} = \frac{1017}{147} = 6{,}92$$

$$E_p = \frac{S_p}{p} = \frac{T_1}{p.T_p} = \frac{6{,}92}{8} = 0{,}86$$

A partir dessas métricas, calculou-se o tempo teórico sequencial e o tempo de execução paralelo para a comparação dos *workflows*. Pela execução da SimiFlow pelo Chiron, o tempo de execução paralelo obtido é de aproximadamente 147 horas, enquanto que o tempo teórico seqüencial é de aproximadamente 1017 horas. Sendo assim, aplicando-se as métricas definidas, obteve-se um valor de *speedup* equivalente a 6,92, enquanto que o valor de eficiência foi igual a 0,86.

O resultado de *speedup* pode ser interpretado como o favorecimento do aumento do número de cores para a diminuição do tempo de execução. Portanto, a proporção máxima possível é igual ao número de processadores dedicados a essa aplicação. Nesse caso, o valor de *speedup* pode ser considerado alto, revelando a considerável capacidade de paralelização desempenhada pelo Chiron.

Da mesma forma, o valor de eficiência representa quanto a adição de um processador foi favorável para a resolução do problema, proposto pela aplicação. O valor de eficiência é representado entre 0 e 1, sendo que valores mais próximos de 1 representam alto aproveitamento da aplicação ao utilizar *p* processadores. Logo, o valor de eficiência obtido revela que o uso de 8 processadores proporcionou o uso de 86% desse tempo solucionando o problema em questão.

A partir dos experimentos apresentados nesse capítulo, pode-se concluir que a arquitetura SimiFlow apresenta um mecanismo eficiente para a identificação de grupos de *workflows* similares, quando considera-se um conjunto pequeno de *workflows*, como o apresentado no primeiro estudo. Contudo, muitos desses *workflows* não foram analisados quanto a sua estrutura, com os outros *workflows* existentes. Além disso, observou-se que o uso do Chiron para realizar a paralelização favoreceu o uso dos processadores com 86% de eficiência, sendo este um resultado positivo, caso considere-se o custo de comunicação e de sincronização.

7. Conclusão

Os *workflows* científicos apresentam-se como uma abordagem de destaque para a modelagem de experimentos científicos baseados em simulações, sendo necessário, para tanto, determinar as características de seus componentes. Contudo, as configurações desses componentes podem se tornar algo muito complexo, em virtude de um aumento significativo no número de programas e parâmetros a serem definidos. Uma alternativa é o uso de diferentes níveis de abstração para a especificação desses *workflows*.

Como poucos sistemas existentes são capazes de proporcionar isso e pelo fato de ser difícil determinar se um *workflow* concreto já modelado pertence a um grupo de *workflows* concretos já definidos (ou seja, pertence a um mesmo experimento), o ideal seria um mecanismo capaz de analisar, comparar e agrupar esses *workflows* concretos. A arquitetura SimiFlow apresenta esse mecanismo como proposta, para proporcionar a comparação e o agrupamento dos *workflows* baseado no cálculo de similaridade. Além disso, essa arquitetura pode favorecer a organização de repositórios de *workflows* concretos, propondo novos grupos de acordo com um grau de similaridade, como por exemplo, no sítio *myExperiment*.

A arquitetura SimiFlow apresenta também uma estrutura expansível, baseada em implementações de cartuchos. Cada cartucho apresenta uma determinada funcionalidade, sendo três os cartuchos desenvolvidos: importação de *workflows*, comparação por similaridade e de agrupamento de *workflows*. Para cada cartucho, diferentes algoritmos podem ser desenvolvidos, respeitando a funcionalidade do mesmo.

O primeiro estudo realizado nesse projeto apresenta uma aplicação de *workflows* científicos utilizados em apenas uma área de conhecimento, a bioinformática (sítio

myExperiment). Pelos resultados obtidos, observa-se que a SimiFlow possui algoritmos de comparação e agrupamento com solução bastante promissora para a formação de grupos de *workflows* científicos.

Esses grupos formados podem ser utilizados na definição de linhas de experimentos, através de uma representação num nível de abstração superior. Além disso, essa forma de representação proporciona, posteriormente, o processo de derivação, favorecendo a obtenção dos *workflows* concretos que deram origem às linhas de experimentos.

Além disso, a realização do segundo estudo experimental constatou que o problema de calcular os valores de similaridades é computacionalmente tratável, quando se utiliza uma quantidade elevada de *workflows*. O mecanismo computacional adotado foi a paralelização, proporcionando que mais de uma comparação entre pares de *workflows* fossem realizadas ao mesmo tempo. Vale ressaltar também que ambos os resultados dos dois estudos experimentais foram apresentados em um artigo publicado no III e-Science Workshop (SILVA *et al*., 2010), sendo que este trabalho foi convidado para ser estendido e publicado em revista (SILVA *et al*., 2011).

Dessa forma, a arquitetura SimiFlow apresenta como trabalhos futuros, a análise dos grupos de *workflows* formados no segundo estudo de caso, apresentado na avaliação experimental, o desenvolvimento de um mecanismo capaz de criação de linhas de experimentos, a partir desses grupos de *workflows* identificados, e a proposição de outros algoritmos para os cartuchos existentes.

Referências Bibliográficas

ALTINTAS, I., BERKLEY, C., JAEGER, E., et al., 2004, "Kepler: an extensible system for design and execution of scientific workflows". In: Scientific and Statistical Database Management, pp. 423-424, Greece.

ANDERSON, E. W., FREIRE, J., KOOP, D., et al., 2007, Provenance Challenge - Vistrails. Disponível em: http://twiki.ipaw.info/bin/view/ Challenge/VisTrails,

BAYUCAN, A., HENDERSON, R. L., JONES, J. P., 2000, "Portable Batch System Administration Guide"

BIRSAN, D., 2005, "On plug-ins and extensible architectures", Queue, v. 3, n. 2, pp. 40-46.

BUNKE, H., SHEARER, K., 1998, "A graph distance metric based on the maximal common subgraph", Pattern Recogn. Lett., v. 19, n. 3-4, pp. 255-259.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., et al., 2006, "VisTrails: visualization meets data management". In: SIGMOD, pp. 745-747, Chicago, Illinois, USA.

CARPENTER, B., GETOV, V., JUDD, G., et al., 2000, "MPJ: MPI-like message passing for Java", Concurrency: Practice and Experience, v. 12, n. 11, pp. 1019-1038.

CAVALCANTI, M. C., TARGINO, R., BAIÃO, F., et al., 2005, "Managing structural genomic workflows using web services", Data & Knowledge Engineering, v. 53, n. 1, pp. 45-74.

CONRADI, R., WESTFECHTEL, B., 1998, "Version Models for Software Configuration Management", ACM Computing Surveys, v. 30, n. 2, pp. 232-282.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al., 2009, Introduction to Algorithms. third edition ed. The MIT Press.

COSTA, B., OGASAWARA, E., MURTA, L., et al., 2009, "Uma Estratégia de Versionamento de Workflows Científicos em Granularidade Fina". In: III e-Science, pp. 49-56, Fortaleza, Ceara, Brazil.

DEELMAN, E., GANNON, D., SHIELDS, M., et al., 2009, "Workflows and e-Science: An overview of workflow system features and capabilities", Future Generation Computer Systems, v. 25, n. 5, pp. 528-540.

FREIRE, J., KOOP, D., SANTOS, E., et al., 2008, "Provenance for Computational Tasks: A Survey", Computing in Science and Engineering, v.10, n. 3, pp. 11-21.

FUSARO, P., VISAGGIO, G., TORTORELLA, M., 1998, "REP - ChaRacterizing and Exploiting Process Components: Results of Experimentation". In: Proceedings of the Working Conference on Reverse Engineering (WCRE'98), pp. 20-29, Honolulu, Hawai.

GAMMA, E., HELM, R., JOHNSON, R., et al., 1994, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

GARG, A., CRITCHLOW, M., CHEN, P., et al., 2003, "An Environment for Managing Evolving Product Line Architectures". In: Proceedings of the International Conference on Software Maintenance, pp. 358-366, Amsterdam, The Netherlands.

GOBLE, C. A., ROURE, D. C. D., 2007, "myExperiment: Social Networking for Workflow-using e-Scientists". In: Proceedings of the 2nd workshop on Workflows in support of large-scale science, pp. 1-2, Monterey, California, USA.

GOBLE, C., WROE, C., STEVENS, R., 2003, "The myGrid project: services, architecture and demonstrator". In: Proc. of the UK e-Science All Hands Meeting

GODERIS, A., DE ROURE, D., GOBLE, C., et al., 2008, "Discovering Scientific Workflows: The myExperiment Benchmarks", IEEE Transactions on Automation Science and Engineering

HAN, J., KAMBER, M., 2006, Data Mining: Concepts and Techniques. Morgan Kaufmann.

HILL, M., 2006, "What is scalability", 28th International Conference on Software Engineering, v. 18, n. 4, pp. 18 - 21.

HULL, D., WOLSTENCROFT, K., STEVENS, R., et al., 2006, "Taverna: a tool for building and running workflows of services", Nucleic Acids Research, v. 34, n. 2, pp. 729-732.

JAIN, A. K., MURTY, M. N., FLYNN, P. J., 1999, "Data clustering: a review", ACM Comput. Surv., v. 31, n. 3, pp. 264-323.

JUNG, J.-Y., BAE, J., 2006, "Workflow Clustering Method Based on Process Similarity", Computational Science and Its Applications - ICCSA 2006, , pp. 379-389.

JWE, 2009, Java Workflow Editor, http://www.enhydra.org.

LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., et al., 2006, "Scientific workflow management and the Kepler system: Research Articles", Concurrency and Computation: Practice and Experience, v. 18, n. 10, pp. 1039-1065.

BAKER, M., CARPENTER, B., SHAFI, AAMIR. MPJ: Enabling Parallel Simulations in Java.

MATTOSO, M., WERNER, C., TRAVASSOS, G. H., et al., 2010, "Towards Supporting the Life Cycle of Large-scale Scientific Experiments", Int Journal of Business Process Integration and Management, v. 5, n. 1, pp. 79–92.

MATTOSO, M., WERNER, C., TRAVASSOS, G., et al., 2009, "Desafios no Apoio à Composição de Experimentos Científicos em Larga Escala". In: SEMISH - CSBC, pp. 307-321, Bento Gonçalves, Rio Grande do Sul, Brazil.

MCPHILLIPS, T., BOWERS, S., ZINN, D., et al., 2009, "Scientific workflow design for mere mortals", Future Generation Computer Systems, v. 25, n. 5, pp. 541-551.

NORTHROP, L. M., 2002, "SEI's software product line tenets", IEEE Software, v. 19, n. 4, pp. 32-40.

OGASAWARA, E., DIAS, J., OLIVEIRA, D., et al., 2011, "An Algebraic Approach for Data-Centric Scientific Workflows", Proceedings of the VLDB Endowment, v. 4, n. 12, pp. 1328-1339.

OGASAWARA, E., PAULINO, C., MURTA, L., et al., 2009, "Experiment Line: Software Reuse in Scientific Workflows", In: Winslett, M. [org.] (eds), Scientific and Statistical Database Management, , chapter 5566, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 264-272.

OHST, D., WELLE, M., KELTER, U., 2003, "Differences between versions of UML diagrams". In: Proc. 9th ESEC, pp. 227-236, Helsinki, Finland.

OINN, T., ADDIS, M., FERRIS, J., et al., 2004, "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, v. 20, pp. 3045-3054.

OINN, T., LI, P., KELL, D. B., et al., 2007, "Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community", Workflows for e-Science, Springer, pp. 300-319.

OLIVEIRA, D., OGASAWARA, E., SEABRA, F., et al., 2010, "GExpLine: A Tool for Supporting Experiment Composition", Provenance and Annotation of Data and Processes, , chapter 6378, Springer Berlin / Heidelberg, pp. 251-259.

DE ROURE, D., GOBLE, C., 2007, "myExperiment – A Web 2.0 Virtual Research Environment.", In International Workshop on Virtual Research Environments and Collaborative Work Environments.

SANTOS, E., LINS, L., AHRENS, J. P., et al., 2008, "A First Study on Clustering Collections of Workflow Graphs", Proc. IPAW 2008, Springer-Verlag, pp. 160-173.

SiDiff, 2010, SiDiff, http://www.sidiff.org.

SILVA, V., CHIRIGATI, F., MAIA, K., et al., 2010, "SimiFlow: Uma Arquitetura para Agrupamento de Workflows por Similaridade". In: IV e-Science, Belo Horizonte, Minas Gerais, Brazil.

SILVA, V., CHIRIGATI, F., MAIA, K., et al., 2011, "Similarity-based Workflow Clustering", Journal of Computational Interdisciplinary Science

STEVENS, R., MCENTIRE, R., GOBLE, C., et al., 2004, "myGrid and the drug discovery process", Drug Discovery Today: BIOSILICO, v. 2, n. 4 (jul.), pp. 140-148.

TAYLOR, I. J., DEELMAN, E., GANNON, D. B., et al., 2007, Workflows for e-Science: Scientific Workflows for Grids. 1 ed. Springer.

UHRIG, S., 2008, "Matching class diagrams: with estimated costs towards the exact solution?". In: Proc. 2008 CVSM, pp. 7-12, Leipzig.

WfMC, I., 2009, Binding, WfMC Standards, WFMC-TC-1023, http://www. wfmc. org, 2000.

WOLSTENCROFT, K., ALPER, P., HULL, D., et al., 2007, "The myGrid ontology: bioinformatics service discovery", Int. J. Bioinformatics Res. Appl., v. 3, n. 3, pp. 303-325.