Workflows Científicos

Laryssa Aparecida Machado da Silva

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientadora: Profa. Regina Maria Maciel Braga Villela

Juiz de Fora, MG
Dezembro de 2007

Workflows Científicos

Laryssa Aparecida Machado da Silva

Monografia submetida ao corpo docente do Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora, como parte integrante dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovada pela banca constituída pelos seguintes professores:

_____

Profa. Regina Maria Maciel Braga – orientadora
DSc. em Engenharia de Sistemas e Computação, COPPE/UFRJ, 2000

_____

Prof. Tarcísio de Souza Lima
MSc. em Informática, PUC/RJ, 1988

_____

Profa. Fernanda Cláudia Alves Campos
DSc. em Engenharia de Sistemas e Computação, COPPE/UFRJ, 1999

Juiz de Fora, MG
Dezembro de 2007

Agradecimentos

A Deus,

por me dar saúde e força para

concluir mais esta etapa da minha vida.

Sumário

Lista de Reduções

BLAST	*Basic Local Alignment Search Tool*
BPEL4WS	*Business Process Workflow Language for Web Services*
CSCW	*Computer Supported Cooperative Work*
DAML-OIL	*DARPA Agent Markup Language - Ontology Inference Layer*
DNA	*Deoxyribonucleic Acid*
GGF	*Global Grid Forum*
GPU	*Graphics Processing Unit*
KAVE	*Knowledge Annotations and Verification of Experiments*
MIR	*myGrid Information Repository*
OASIS	*Organization for the Advancement of Structured Information Standards*
OGSA	*Open Grid Service Architecture*
OGSA-DAI	*Open Grid Service Architecture – Data Access and Integration*
OGSA-DQP	*Open Grid Service Architecture – Distributed Query Processing*
OGSI	*Open Grid Service Infrastructure*
RNA	*Ribonucleic Acid*
SCUFL	*Simple Conceptual Unified Flow Language*
SGW	Sistema de Gerência de *Workflows*
SOA	*Service-oriented Architecture*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
VB	*Vistrail Builder*
VCM	*Vistrail Cache Manager*
VP	*Vistrail Player*
VS	V*isualization Spreadsheet*
WfMC	*Workflow Management Coalition*
WSDL	*Web Services Description Language*
WSFL	*Web Services Flow Language*
XML	*Extensible Markup Language*

Lista de Figuras

Resumo

O termo *e-science* foi introduzido na comunidade científica com o objetivo de englobar as tecnologias necessárias ao suporte à pesquisa colaborativa e multidisciplinar que emergiu em vários campos da ciência. As implicações sociais deste novo cenário com tecnologias que permitem o crescimento da ciência colaborativa consistem em uma popularização na utilização das tecnologias e ferramentas de *e-science* por diferentes comunidades de pesquisa.

Essa nova infra-estrutura de conhecimento distribuído que está emergindo faz uso das tecnologias de informação e comunicação para o armazenamento de dados científicos, sua análise e o apoio ao trabalho colaborativo. No contexto de *e-science*, a mobilidade e a facilidade propiciadas pelos dados digitais favorece o crescimento da qualidade, produtividade e confiabilidade das pesquisas. A comunicação por redes permite a ligação entre grupos de pesquisadores, sejam eles da mesma área científica ou não e pertencentes ou não a um mesmo país, facilitando a comunicação e colaboração entre esses grupos. Sendo aplicada principalmente em áreas científicas, que produzem grandes volumes de dados, como física de partículas, astronomia e genética, esta nova infra-estrutura vem sendo desenvolvida para permitir que as pesquisas científicas possam ser feitas em larga escala, com maior eficiência e explorando questões cada vez mais complexas.

Neste cenário, são apresentadas as principais tecnologias relacionadas a *e-science*, com especial ênfase em *workflows* científicos, englobando questões como proveniência de dados, e serviços *Web.* O *workflow* científico é utilizado para representar experimentos científicos. Os serviços *web* podem ser utilizados como componentes de *worflows* científicos, permitindo assim, o reuso e compartilhamento desses *workflows*. A proveniência de dados permite ao cientista armazenar dados e metadados relacionados às execuções dos *workflows* que representam seus experimentos.

É dispensada maior atenção aos *workflows* científicos por ser essa tecnologia a mais utilizada atualmente para a obtenção de melhores e mais eficientes resultados em variadas áreas de conhecimento. São apresentadas as principais características dos *workflows* científicos, uma linguagem para definição de *workflows* e algumas ferramentas para gerência de *workflows* científicos. Apresenta-se assim, um panorama geral do contexto atual de *e-science*.

Palavras-Chave

Workflow, *e-science*, *grid,* proveniência de dados, rastreabilidade.

Capítulo 1

Introdução

A tecnologia da informação vem revolucionando o modo como diversas áreas da ciência vêm sendo conduzidas. A utilização de recursos computacionais no desenvolvimento da pesquisa beneficia o trabalho das comunidades científicas facilitando o compartilhamento de dados e serviços computacionais, além de contribuir para a construção de uma infra-estrutura de dados e de uma comunidade científica distribuída (LUDÄSCHER *et al.*, 2004). Este contexto em que a computação se torna parte integrante e imprescindível para o sucesso na realização de pesquisas científicas das mais variadas áreas, é o contexto de *e-science*, em que a ciência é realizada com o apoio computacional, se tornando assim, mais eficiente. Soluções computacionais são muito importantes para o desenvolvimento de pesquisas em áreas como a biologia, que gera grande volume de dados e necessita do apoio computacional para a realização e gerenciamento de experimentos.

As atividades de *e-science* estão crescendo por todo o mundo, acompanhadas por uma proliferação de dados e ferramentas. Isto traz novos desafios, por exemplo, como entender e organizar esses recursos, como compartilhar e reusar experimentos bem sucedidos (ferramentas e dados), e como prover interoperabilidade entre dados e ferramentas de diferentes locais e utilizados por usuários com perfis distintos (DIGIAMPIETRI, 2007). Estes desafios geram pesquisas na área da computação, no sentido de desenvolver mecanismos que ofereçam todo o apoio necessário à pesquisa. A computação está em busca de uma infra-estrutura que permita projetar, reusar, anotar, validar, compartilhar e documentar (DIGIAMPIETRI, 2007).

Os *workflows* científicos são mecanismos para representar computacionalmente os experimentos científicos. Os cientistas estão interessados em ferramentas que tragam o poder das bases de dados distribuídas, e outros recursos computacionais disponíveis em *grids* ou na *Web*, para sua máquina local e permita que eles as utilizem para executar seus *workflows* científicos (LUDÄSCHER *et al.*, 2004; DIGIAMPIETRI, 2007). *Workflows* científicos estão sendo cada vez mais adotados como meios para especificar e coordenar a execução de experimentos que envolvem participantes em locais distintos. Eles permitem a representação e execução de tarefas que usam dados e ferramentas heterogêneos. Seu uso está se disseminando na área de *e-science*, com aplicação em biologia, física, química ou astronomia, por exemplo. A bioinformática, foi um dos primeiros domínios científicos a adotar tais *workflows,* utilizado-os em experimentos *in silico* (DIGIAMPIETRI, 2007).

Outro aspecto importante está ligado à proveniência dos dados e ferramentas utilizadas em cada experimento. "A proveniência de dados, também chamada de linhagem, genealogia ou *pedigree* é a descrição das origens de um item de dado e do processo pelo qual foi produzido. A proveniência dos dados auxilia a formar uma visão da qualidade, da validade e de quão recente é a informação" (BUNEMAN, 2001 *in* BALBI; PIRES e MATTOSO 2005). Este tipo de informação é fundamental para que um experimento possa ser reproduzido, além de assegurar sua qualidade (BALBI; PIRES e MATTOSO, 2005; DIGIAMPIETRI, 2007).

Além da necessidade de funcionalidades de proveniência, os sistemas de *workflows* devem prover mecanismos de rastreabilidade para que o usuário possa analisar detalhes dos experimentos já executados. Isto requer ferramentas para re-execução de partes dos experimentos, consultas sobre resultados parciais e parâmetros utilizados etc. As funcionalidades dos sistemas não são suficientes por si só, pois o cientista tipicamente não possui muitos conhecimentos em desenvolvimento de software e, desta forma, precisa de ferramentas que simplifiquem a especificação e execução de experimentos (DIGIAMPIETRI, 2007).

1.2 Motivação e objetivos

Soluções computacionais são muito importantes para possibilitar o desenvolvimento de pesquisas científicas em diversas áreas de conhecimento. Os recursos computacionais beneficiam o trabalho das comunidades científicas, facilitando o compartilhamento de dados e serviços computacionais. Pesquisas de grande magnitude, como os estudos sobre o genoma, por exemplo, são realizadas com a cooperação de diversos grupos científicos, separados geograficamente, compartilhando dados e ferramentas entre si. Dessa forma, é interessante conhecer um pouco sobre os recursos e tecnologias computacionais utilizados no meio científico, que possibilitam a realização dessas pesquisas que podem gerar muitos benefícios à comunidade como um todo.

Neste contexto, este trabalho busca analisar o atual estado da arte de *e-science*, que consiste na pesquisa científica realizada com o auxílio de recursos computacionais. O principal mecanismo utilizado pelos cientistas para automatizar os procedimentos experimentais e, assim, melhorar o desempenho de suas pesquisas, é o *workflow* científico. Assim, temos o objetivo de analisar as principais características dessa tecnologia, fazendo sua ligação com o contexto de *e-science*, fazendo um paralelo entre este tipo de *workflow* e

o tipo mais tradicional, conhecido como *workflow* de negócio, além de conhecer e analisar uma linguagem e algumas ferramentas que auxiliam na construção e execução dos *workflows* científicos.

1.3 Organização da monografia

A presente monografia está dividida da seguinte forma, além desta introdução: o capítulo 2 aborda os principais conceitos relacionados ao contexto de *e-science*, propiciando um entendimento geral e mostrando os principais fatores que estão envolvidos no assunto; o capítulo 3 apresenta o conceito de *workflow,* partindo da visão do *workflow* científico que está inserido no contexto de *e-science,* mostrando a origem deste conceito e apresentando alguns aspectos de uma linguagem para a definição de *workflows*; o capítulo 4 apresenta, de forma resumida, as principais funcionalidades e características de algumas ferramentas para a definição, execução e gerenciamento de *workfllows*; e, finalmente, o capítulo 5 apresenta conclusões que podem ser tiradas a partir do conhecimento adquirido com o presente trabalho.

Capítulo 2

Fundamentação teórica para *e-science*

Nas mais diversas áreas da ciência são aplicados cada vez mais recursos computacionais para o auxílio ao desenvolvimento de atividades de pesquisa. São utilizados dados científicos oriundos de programas, arquivos, instrumentos, sensores, experimentos etc. Algumas áreas de pesquisa, que geram grande quantidade de dados, utilizam recursos computacionais de alto desempenho para atender à sua demanda de armazenamento, processamento e análise dos dados obtidos (LAUSCHNER, 2005).

É importante guardar toda a grande quantidade de informação gerada nos centros de pesquisa para evitar que dados importantes sejam perdidos e que o trabalho tenha que ser repetido. Para permitir que os cientistas mantenham o foco em suas pesquisas e não se preocupem com questões logísticas, como o armazenamento e a recuperação de informações, são utilizadas tecnologias como a proveniência de dados, *workflows* científicos e serviços *Web* para a construção de ferramentas de apoio a experimentos científicos (BALBI; PIRES e MATTOSO , 2005).

O termo *e-science* foi introduzido, no Reino Unido, pelo então diretor geral dos conselhos de pesquisa, *Sir* John Taylor, para encapsular as tecnologias necessárias ao suporte à pesquisa colaborativa e multidisciplinar que emergiu em vários campos da ciência (HINE, 2006). Este termo é geralmente empregado para descrever o desenvolvimento de infra-estruturas de serviços de software capazes de prover acesso a facilidades remotas, recursos computacionais remotos, armazenamento de informações em bancos de dados dedicados e disseminação e compartilhamento de dados, resultados e conhecimento. Em essência, "um ambiente de *e-science* deve permitir o compartilhamento de recursos coordenados em larga-escala entre comunidades dinâmicas de indivíduos, grupos, laboratórios e instituições, permitindo o gerenciamento cooperativo de facilidades (equipamentos, instrumentação, experimentos etc.) e análise colaborativa de produtos (dados) oriundos dessas facilidades" (e-s05a, e-s05b *in* LAUSCHNER, 2005).

São características de *e-science* o acesso a uma vasta coleção de dados, utilização de recursos computacionais em larga escala, utilização de recursos heterogêneos e dinâmicos de múltiplas organizações e *workflows* (CARDOSO; MATTOSO e MATTOS, 2007).

Segundo CARDOSO, MATTOSO e MATTOS (2007), em uma aplicação de *e-science,* pode ser definido um ciclo de vida para experimentos científicos. Esse ciclo é composto por fases, que ajudam no desenvolvimento e estruturação das ferramentas computacionais que irão auxiliar na realização dos experimentos, são elas:

- a criação do experimento, de acordo com a área a qual ele pertence e os objetivos que se pretende alcançar a partir desse experimento;
- a personalização do experimento, ou seja, a definição computacional para execução de tal experimento;
- a execução do experimento e seu monitoramento;
- a geração de dados de proveniência, que serão úteis para possível re-execução ou reprodução do experimento futuramente, e a obtenção dos resultados do experimento;
- o compartilhamento dos resultados obtidos com outros cientistas;
- a descoberta, a partir de dados de proveniência e *workflows* científicos, de experimentos que já foram realizados e a reutilização desses experimentos e integração de seus dados.

Ao longo desse ciclo de vida são utilizadas diversas tecnologias e ferramentas computacionais. É apresentada neste capítulo uma visão geral destas tecnologias.

2.1 Proveniência de dados

Workflows consistem em modelos para a execução controlada de múltiplas tarefas. Para a representação de experimentos científicos, é utilizado um tipo específico chamado de *workflow* científico[1], que é mais flexível e permite modificações em tempo de real, além de tratar possíveis exceções durante sua execução (DIGIAMPIETRI, 2007). Para possibilitar que os processos científicos sejam mapeados para *workflows* com maior eficiência e precisão, são necessários sistemas para gerenciar os *workflows*, e que auxiliem os pesquisadores na definição, validação, otimização e execução dos mesmos. Esses sistemas devem ser capazes de permitir a modelagem dos *workflows* correspondentes aos processos científicos e também devem permitir coordenar a sua execução, controlando a utilização de recursos.

[1] *Workflow* científicos serão detalhados no capítulo 3.

Além do gerenciamento de *workflows* são necessários sistemas de gerência de dados, que tratam do armazenamento e da manipulação dos dados envolvidos nesse processo. A gerência dos dados de um *workflow* científico também envolve a captura e armazenamento de dados relevantes para as pesquisas, que são gerados ao longo do processo de execução de *workflows*. Estes dados são denominados dados de proveniência (LEMOS, 2004). A captura desses dados é complexa, pois envolve fatores específicos de cada área científica que utiliza os sistemas de computação para a realização de pesquisas. Cada domínio de aplicação científica possui esquemas, metadados ou ontologias específicos, ou seja, os dados de proveniência importantes para pesquisas de bioinformática, por exemplo, poderão não ser importantes para outras áreas científicas (MATTOS e MATTOSO, 2007). Os dados referentes a experimentos *in silico*[2] sofrem freqüentes atualizações no decorrer das etapas de pesquisa realizadas. Bancos de dados genômicos, por exemplo, são frequentemente atualizados. Por causa destas características o reuso deste conhecimento gerado garante grande economia de recurso e tempo, além de facilitar a definição da adequabilidade dos resultados ao problema que está sendo investigado e permitir que a origem dos resultados seja identificada (BALBI; PIRES e MATTOSO, 2005).

Para que seja possível ter informações precisas sobre a origem dos dados, é necessário o trabalho de proveniência de dados[3]. O dado de proveniência é um tipo de metadado que descreve o processo dos experimentos. Dentre esses dados podem estar o propósito do experimento, seus resultados e anotações importantes e esclarecedoras feitas pelo cientista realizador do experimento, além de dados de qualidade e temporais sobre os resultados do experimento (CARDOSO; MATTOSO e MATTOS, 2007).

Os dados científicos têm menor valor (menor confiabilidade) se não for possível que outros cientistas verifiquem a origem ou proveniência dos mesmos. A proveniência é essencial para que os experimentos sejam validados e verificados por outros cientistas ou mesmo aqueles que realizaram o experimento anteriormente (CARDOSO; MATTOSO e MATTOS, 2007).

---

[2] A expressão *in silico* é, originalmente, usada para denotar simulações computacionais que modelam um processo natural ou de laboratório (LEMOS, 2004).

[3] "A proveniência de dados, também chamada de linhagem, genealogia ou *pedigree* é a descrição das origens de um item de dado e do processo pelo qual foi produzido. A proveniência dos dados auxilia a formar uma visão da qualidade, da validade e de quão recente é a informação" (BUNEMAN, 2001 *in* BALBI; PIRES e MATTOSO, 2005).

Com relação aos dados gerados a partir de experimentos científicos realizados com o auxílio de ferramentas computacionais, é necessário um suporte ao armazenamento de metadados, além dos próprios dados oriundos de resultados experimentais, que irão permitir que as ferramentas de monitoramento de proveniência de dados ofereçam, com maior eficiência, serviços importantes como o de busca de recursos que facilitam o trabalho dos cientistas. O modelo de proveniência que se obtém deve ser claramente definido e relevante ao *workflow* definido para cada processo experimental, pois a proveniência é um processo que requer a cooperação de várias partes envolvidas e a clareza deste processo vai influenciar na confiança que os cientistas terão na qualidade dos dados gerados (BALBI; PIRES e MATTOSO, 2005).

Porém, a maioria dos dados disponíveis para a comunidade científica tem pouca informação de proveniência, pois falta uma infra-estrutura adequada de apoio aos experimentos e o conceito de proveniência é relativamente novo e pouco explorado (BALBI; PIRES e MATTOSO, 2005).

Segundo CARDOSO, MATTOSO e MATTOS (2007), existem dois tipos de dados de proveniência:

- *Annotations*: anotações padrão como data de criação de um objeto, última atualização, quem é o dono, qual o formato do objeto, etc.
- *Derivation Path*: dados relacionados com a execução do *workflow*. Relaciona os dados de entrada (*input data*) e os resultados obtidos a partir deles, gravando estas relações.

Além de referenciarem diferentes tipos de dados de proveniência, alguns autores fazem distinção entre os tipos de proveniência *why provenance*, referente a dados que exercem alguma influência no resultado, e *where provenance*, referente aos dados usados para gerar os resultados (BALBI; PIRES e MATTOSO, 2005).

É cada vez mais importante que as ferramentas de monitoramento de proveniência de dados sejam desenvolvidas utilizando-se padrões e linguagens abertas para que a interação entre as diversas ferramentas seja possível. Na construção de ferramenta de monitoramento de proveniência dos dados se busca o armazenamento eficiente dos dados de proveniência para possibilitar o rastreamento das origens dos dados e sua visualização, a determinação de diferenças entre execuções de um experimento e a reprodução deste experimento (BALBI; PIRES e MATTOSO, 2005).

2.2 Integração de informações

Atualmente, as pesquisas científicas são, em sua maioria, realizadas com o apoio de vastos conjuntos de ferramentas computacionais. A colaboração com dados, idéias e experimentos, em tempo real, entre pesquisadores separados geograficamente também é cada vez mais difundida e ocorre com grande freqüência no cenário de pesquisas atual. Este cenário demanda o desenvolvimento de novas tecnologias, capazes de permitir a integração de informações geradas por diferentes grupos de pesquisa (MATTOS e MATTOSO, 2007).

Para tornar possível a integração entre as informações obtidas a partir de pesquisas feitas por grupos diferentes, é importante promover a integração das bases de dados utilizadas por cada um desses grupos para armazenar seus dados. Os principais objetivos da integração de bases de dados são a obtenção de uma visão única sobre essas diversas bases e o intercâmbio de dados de uma base para a outra. Para acessar várias bases de dados como uma única base é necessário relacionar dados entre bases diferentes, independente do fabricante ou dos esquemas de dados da base. Para que esse acesso seja possível é feito o uso de sistemas de integração ou mediadores (KROTH, 2003). Na Figura 1 temos um esquema geral da integração entre bases de dados distintas.

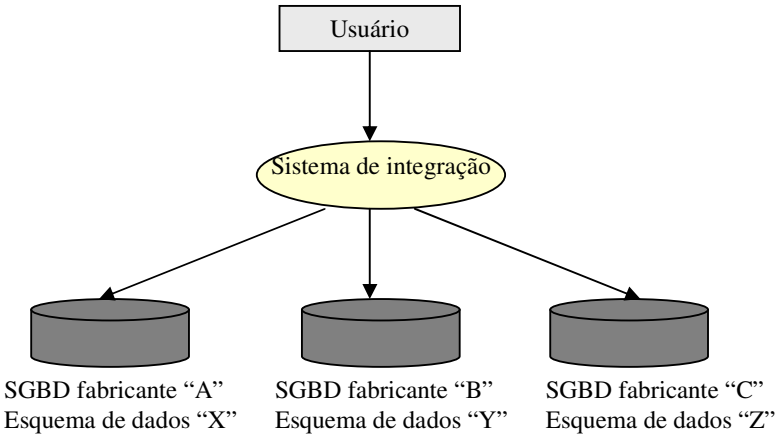

Figura 1 – Integração de bases de dados (KROTH, 2003)

Com os avanços tecnológicos, estão sendo desenvolvidos novos paradigmas de computação que, por sua vez, implicam em novas aplicações, desenvolvidas de acordo com esses paradigmas para atender às necessidades tecnológicas. As necessidades das novas aplicações com alcance cada vez mais multidisciplinar, ou seja, aplicações mais genéricas, que possam ser utilizadas em diversas áreas científicas, implicam no

desenvolvimento de uma tecnologia de bancos de dados robusta e flexível que possa ser empregada em diversos ambientes e para inúmeros propósitos. A necessidade de bancos de dados aplicáveis a ambientes diversos vem do fato de que os sistemas atuais necessitam de interação com outros sistemas e, muitas vezes, se aplicam a ambientes distribuídos. Além disso, os dados podem ser compartilhados e, portanto, seu formato deve ser admitido e entendido em outros ambientes que não são seu ambiente original (MATTOSO, 2007).

Uma área que necessita de armazenamento de grande quantidade de dados, e de compartilhamento de dados é a bioinformática. A bioinformática consiste em técnicas de matemática, estatística e ciência da computação, aplicadas à resolução de problemas biológicos (DIGIAMPIETRI, 2007). "Um banco de dados biológico constitui um grande conjunto de dados persistentes, geralmente associado a um software projetado para atualizar, consultar e recuperar componentes dos dados armazenados no sistema". (BIOINFORMATICS FACTSHEET, 2004 *in* OLIVEIRA e OLIVEIRA, 2006).

Em geral, os dados biológicos são disponibilizados na *Web* para que possam ser compartilhados com outros cientistas. Porém, muitas das bases de dados disponibilizadas na *Web* foram construídas pelos próprios biólogos que realizam as pesquisas. Como cada grupo de pesquisa constrói sua própria base de dados, a integração entre esses vários resultados obtidos fica dificultada, pois não há padronização de taxonomia e não há a adoção de um vocabulário comum entre os diversos grupos de pesquisa, o que leva à utilização de termos diferentes para conceitos iguais e utilização de conceitos diferentes para termos iguais (KROTH, 2003). Para resolver este tipo de problema, seria importante um esforço multidisciplinar para a definição de padrões de nomenclatura e desenvolvimento de ferramentas capazes de reconhecer esses padrões e prover a integração entre os dados de diferentes origens.

Alguns mecanismos que podem ser utilizados para a integração de diferentes bancos de dados são a análise de requisitos comuns como, a autonomia das bases de dados (de comunicação, de projeto, de execução, de associação), heterogeneidade (nem todos os requisitos são atendidos por todas as bases) nos níveis conceitual (conceitos diferentes), físico (paradigmas diferentes) e de implementação (tipos de dados diferentes). Também são observados requisitos característicos para bases de dados biológicas como a evolução freqüente dos esquemas das bases de dados, dificuldade no entendimento do escopo de uma base de dados, conflitos de semântica entre as bases de dados e esquemas de dados não disponíveis (KROTH, 2003).

Mesmo com a integração a autonomia das bases de dados integradas deve ser mantida, sobretudo quando a integração se dá entre bases de origens distintas que, mesmo integradas, continuarão também sendo usadas separadamente. Cada base de dados deve continuar possuindo controle independente sobre seus dados. Essas bases devem passar a apresentar a disposição para compartilhar dados desde que o controle individual seja mantido. Para que o compartilhamento ocorra com esta autonomia, é importante manter as autonomias de projeto, comunicação, execução e associação (KROTH, 2007).

O esquema de bases de dados federados, que apresenta uma arquitetura composta por bases de dados que disponibilizam dados sem a necessidade de uma integração de esquema, pode ser considerado um dos melhores para a integração entre bases de dados científicos. A arquitetura de bases de dados federados permite mais autonomia de associação entre bases de dados independentes em um ambiente cooperativo, fator que é muito importante no compartilhamento de dados científicos. Além disso, este esquema apresenta maior flexibilidade por apresentar um esquema externo modelado de acordo com os requisitos do usuário externo (KROTH, 2007).

A fase inicial do estudo de genomas é um exemplo de necessidade de integração entre diversas bases de dados. Essa fase começa com a obtenção de seqüências de DNA, sem nenhum significado biológico, em laboratório. Para analisar seqüências de DNA, RNA e proteínas e obter informações biológicas relevantes, os pesquisadores utilizam diversas ferramentas computacionais e têm necessidade de armazenar grande volume de informações em bases de dados biológicas (LEMOS, 2004).

Cada célula de um organismo vivo contém cromossomos, que são compostos de uma seqüência de pares de bases de DNA. O DNA é uma seqüência linear de quatro nucleotídeos (também chamados de bases, representados pelos caracteres *A*, *T*, *C* e *G*), que é a fonte básica da informação genética. Para obtenção de uma seqüência de DNA, um genoma é fragmentado e submetido a um seqüenciador automático. Existem algoritmos, chamados de algoritmos de alinhamento, que buscam unir os fragmentos a fim de construir uma seqüência completa. A ferramenta computacional mais usada em Biologia, para fazer a busca de seqüências similares, é o algoritmo BLAST - *Basic Local Alignment Search Tool* (Altschul, 1998; Altschul, 1990; WU-BLAST, 2004a; Lemos, 2000a, b; Casanova, 2001; Lemos, 2003b *in* LEMOS, 2004), que serve para realizar consultas a bancos de dados na *Web* em busca de similaridade entre as seqüências encontradas e seqüências já identificadas e disponibilizadas nestes bancos de dados (KROTH, 2003).

Como podemos observar, a área de bioinformática, assim como diversas outras áreas científicas, além de gerar grandes volumes de dados que precisam ser armazenados, também necessita de integração entre dados de diferentes origens. “O crescente volume e a distribuição das fontes de dados e a implementação de novos processos em bioinformática facilitaram enormemente a fase de análise dos dados obtidos, porém criaram uma demanda por ferramentas e sistemas semi-automáticos para lidar com tal volume e complexidade” (LEMOS, 2004).

Algumas tecnologias, que são largamente utilizadas nas áreas científicas, representam desafios que estão direcionando a pesquisa e desenvolvimento de novas tecnologias de bancos de dados (MATTOSO, 2007). Alguns exemplos dessas tecnologias são arquiteturas ponto a ponto, *grids*, serviços *Web*, computação pervasiva e ubíqua, sensores em redes, bibliotecas virtuais digitais, e comunidades virtuais, entre outras.

2.3 Serviços *Web*

As ferramentas de *e-science* são muito utilizadas em pesquisas ligadas à área da bioinformática. Para a realização de experimentos *in silico,* por exemplo, múltiplas combinações de recursos e programas são utilizadas. A utilização de serviços *Web* é bastante propícia nesta situação devido às suas características de interoperabilidade, escalabilidade e flexibilidade (BALBI; PIRES e MATTOSO, 2005).

O serviço *Web* promove interoperabilidade pela minimização dos requisitos para a compreensão entre um provedor e um requisitante de serviço, porque possui descrição da interface, descrição do serviço, protocolo de colaboração e negociação (BAX e LEAL, 2001). Suas características de escalabilidade e flexibilidade vêm do fato de que este tipo de serviço oferece independência de plataforma e de linguagem, possibilita integração *just-in-time* (ligação (*binding*) dinâmica) entre os diversos serviços, possibilita a existência de sistemas auto-configuráveis, adaptáveis e robustos. O serviço *web* reduz a complexidade pelo encapsulamento, ou seja, cada serviço tem uma interface e uma descrição muito bem definidas e pode ser invocado, a partir dessas informações, por qualquer sistema sem que este tenha que conhecer detalhes sobre seu funcionamento (BAX e LEAL, 2001).

A integração entre serviços, chamada *just-in-time,* é possibilitada pela comunicação inter-componentes que, implementada com XML, permite que os detalhes das interfaces de comunicação entre os processos sejam descobertos em tempo de execução (ou conexão), ao invés de especificados rigidamente em tempo de concepção.

Este fato aumenta substancialmente a flexibilidade das aplicações (Shah, 2001 *in* BAX e LEAL, 2001).

Um serviço *Web* tem uma interface descrita em um formato processável pela máquina (WSDL - *Web Services Description Language*) que é utilizada para interação com outros serviços *Web* através de mensagens SOAP (*Simple Object Access Protocol)* (W3C, 2001 *in* BAX e LEAL, 2001).

A linguagem WSDL é uma linguagem baseada em XML para especificação das propriedades de um serviço *Web*, tais como o que ele faz, onde está localizado e como realizar uma chamada a ele (BAX e LEAL, 2001). A descrição semântica do comportamento do serviço deve ser legível ao ser humano, mas também permitir busca automática e uso por outros serviços.

Para funcionarem bem em conjunto, de forma flexível e dinâmica, os serviços precisam compartilhar princípios organizacionais que, juntos, constituem uma Arquitetura Orientada a Serviço - "*Service-oriented Architecture*" (SOA). Esta arquitetura focaliza-se em como os serviços são descritos e organizados dinamicamente, propiciando descoberta e uso automáticos (BAX e LEAL, 2001). A figura 2 mostra o esquema de funcionamento da arquitetura SOA.

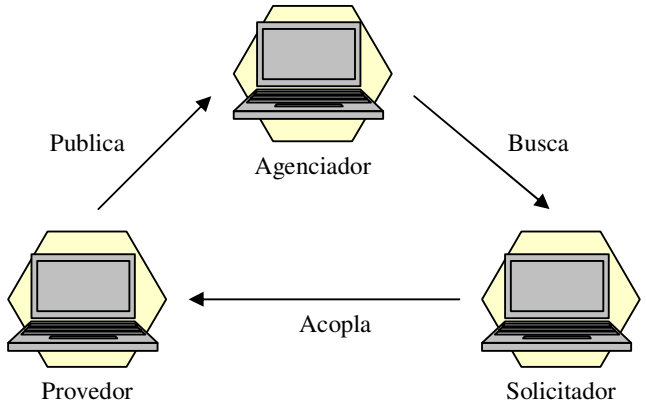

Figura 2 - Interações em uma arquitetura orientada a serviços (BAX e LEAL, 2001)

O provedor (*provider*) de serviços publica a disponibilidade de seus serviços e responde a requisições de uso dos serviços publicados. A operação de publicação (*publish*) permite ao provedor registrar suas habilidades e seus requisitos de interface junto a um agenciador de serviços (*broker*), que registra e categoriza os provedores de serviços e

oferece serviços de pesquisa. A operação de busca (*Find*) permite que o solicitador encontre o serviço desejado junto ao agenciador. Um solicitador (*requestor*) usa um agenciador para encontrar o serviço que precisa e conecta-se ao provedor para utilizar o serviço. A operação de acoplamento (*Bind*) permite ao solicitador do serviço usar o serviço encontrado (BAX e LEAL, 2001).

Para suportar estas operações, a arquitetura SOA requer descrições de serviços, que se apresentam no formato WSDL. Estas descrições especificam características semânticas do serviço e do provedor do serviço. O agenciador usa tal informação para categorizar o serviço e o solicitador a usa para casar seus requisitos com os do provedor e do serviço.

Algumas características dos serviços *Web* dificultam um pouco sua utilização em aplicações complexas como as de *e-science.* Dentre essas, podemos citar:

- Os serviços *Web* são *stateless* por não terem a possibilidade de guardar os resultados obtidos entre uma solicitação ou invocação e outra seguinte. Se for executada uma seqüência de operações dependentes entre si, é necessário guardar o resultado de uma operação e enviar como parâmetro para a próxima. Porém, em aplicações científicas, em geral, é desejável um tipo de serviço chamado *statefull,* que permita que se possa fazer várias operações complexas a partir de uma solicitação única da aplicação (LAUSCHNER, 2005).
- Os seviços *web* também têm a característica de serem não-transitórios (*non-transient*), ou seja, eles prolongam a vida dos seus clientes e, além disso, são persistentes, pois uma vez iniciados e disponíveis, só ficarão indisponíveis se forem desligados ou bloqueados. Isso faz com que depois que um cliente acessa um serviço *web,* todas as informações possam ser acessadas por um próximo cliente e isso pode causar danos às operações realizadas pelo cliente que acessou primeiro (LAUSCHNER, 2005).

Apesar de existirem problemas quanto ao uso de serviços *Web* para aplicações complexas, existem as vantagens em se utilizar esses serviços, como por exemplo, a padronização, que já foi citada anteriormente, facilitando a interação entre diferentes ferramentas e bases de dados, utilizadas por grupos distintos de pesquisadores que necessitam de se comunicar e compartilhar conhecimento.

Neste contexto, com relação às ferramentas de monitoramento de proveniência de dados, é cada vez mais importante que essas sejam desenvolvidas utilizando-se padrões e linguagens abertas para que a interação entre as diversas ferramentas seja possível (BALBI;

PIRES e MATTOSO, 2005). A construção de ferramentas nos modelos de serviços *Web*, facilita desta forma o compartilhamento de conhecimentos entre grupos distintos.

Pesquisas sobre genoma, por exemplo, utilizam dados disponibilizados na *Web* por outros cientistas. Nesta situação, podemos perceber que a utilização de serviços *Web* é bastante útil, pois permite tanto a disponibilização de dados, como a utilização desses dados por outras pessoas, contribuindo para a evolução mais eficiente das pesquisas.

Além de o serviço Web ser utilizado para compartilhamento de dados na *Web*, seu modelo também serve de base para o compartilhamento de dados, serviços e recursos em ambientes de *grids* computacionais. Um *grid* computacional trata do compartilhamento de recursos relacionados e soluções de problemas, em organizações virtuais distribuídas (LAUSCHNER, 2005).

É possível notar que a estrutura dos serviços *Web* está ligada à estrutura de *grids*, e ambas estão relacionadas às atividades de *e-science*, por permitirem integração de dados, combinação de recursos computacionais heterogêneos e comunicação entre diversas organizações. As características de interoperabilidade, escalabilidade e flexibilidade oferecidas pelos serviços *web* são muito importantes para a integração do conhecimento científico gerado que possibilita, principalmente, a evolução mais confiável e eficiente da ciência nos dias atuais.

2.4 *Grids* computacionais

No contexto de *e-science*, temos a utilização de cada vez mais recursos computacionais para o auxílio ao desenvolvimento de atividades de pesquisa. Muitas vezes, é necessária a combinação desses recursos para obtenção de maior poder de processamento, para a interação entre grupos de pesquisas separados geograficamente, ou simplesmente para se obter resultados finais que precisam ser gerados a partir da agregação de resultados parciais obtidos pelo uso de diversas ferramentas.

A Computação em Grade ou *Grid* Computacional (*Grid Computing*) é uma tecnologia bastante relacionada às atividades de *e-science*. Um *Grid* Computacional trata do compartilhamento de recursos relacionados e soluções de problemas, em organizações virtuais distribuídas geograficamente. “A chave deste conceito é a habilidade de negociar compartilhamento de recursos dispostos ao longo de um conjunto de partes participantes, provedores e consumidores, e então usar os resultados desta fusão de recursos para algum propósito” (LAUSCHNER, 2005).

É importante a facilidade de troca de informações entre grupos de cientistas, pois os dados científicos apresentam-se em grande escala e, em sua maioria, são altamente distribuídos. Normalmente, nenhuma organização possui todos os dados, o que leva à necessidade de integração entre diversas organizações, suas medidas de pesquisa, seus modelos utilizados para a realização dessas pesquisas, etc. Este cenário nos remete à utilização da tecnologia de computação em *grid,* que permite essa integração (CARDOSO; MATTOSO e MATTOS, 2007).

A infra-estrutura oferecida pela computação em *grid* permite que instituições de pesquisa geograficamente distribuídas interajam e compartilhem dados, metadados e modelos através de ferramentas para a execução em ambientes de alto desempenho. Cada componente da grade pode atuar tanto como cliente como na condição de fornecedor de dados, informações e conhecimentos. Essas facilidades permitem maior integração de informações oriundas de diferentes grupos científicos. A computação em grade permite o desenvolvimento e a implementação de infra-estruturas de gerenciamento e disponibilização contínua de dados, informações e conhecimentos relativos ao desenvolvimento e a aplicação de soluções dinâmicas em diversas áreas (LAUSCHNER, 2005).

Porém, existem alguns problemas relacionados à utilização de *Grids* que impedem que o uso deste recurso se torne mais popular na comunidade científica. O uso efetivo das ferramentas de computação em grade é difícil. As ferramentas disponíveis são de utilização complexa, de difícil instalação e administração e são, em sua maioria, muito pesadas (LAUSCHNER, 2005).

Muitas ferramentas de *Grid* são monolíticas, ou seja, tentam prover vários serviços, característicos deste tipo de ferramenta, em uma única unidade. Este tipo de abordagem pode ser muito ruim para um usuário que necessita utilizar somente uma pequena parte destes serviços oferecidos (LAUSCHNER, 2005). No contexto de *e-science*, para um usuário que oferece um único serviço, seria mais eficiente utilizar somente a parte que interessa a este usuário da instalação de um G*rid* Computacional. Porém, utilizando um sistema monolítico, seria necessário envolver todas as características computacionais da Grade, o que seria muito mais complexo consumindo maior tempo do usuário.

Um outro problema, que dificulta a utilização de *Grids* é a existência de um processo muito longo e complexo para a configuração do *middleware,* de um *Grid* Computacional, pois, além dos arquivos de configuração que têm que ser editados, é necessário configurar certificados de usuários e mapear cada máquina envolvida no

sistema. Seria eficiente a utilização de ferramentas para facilitar este trabalho, pois, frequentemente, todas as máquinas envolvidas em um projeto têm sua configuração modificada e, sem o auxílio de uma ferramenta desta natureza, estas modificações têm que ser feitas manualmente, gerando uma carga de trabalho grande e desnecessária (LAUSCHNER, 2005).

Por fim, os usuários se deparam com a dificuldade de instalar os pacotes de ferramentas de grade que são, em geral, muito mal documentados com relação à instalação das diversas ferramentas específicas que eles contêm. A escassa documentação que existe, normalmente está em linguagem apropriada a programadores e desenvolvedores de *Grid*, e não são úteis para os pesquisadores, que são os usuários finais das ferramentas (LAUSCHNER, 2005).

Diante deste cenário cheio de obstáculos, se conclui que para construir sistemas de Computação em Grade úteis e fáceis de serem utilizados em aplicações científicas seria necessário projetar ferramentas mais leves e trabalhar em conjunto com a comunidade de *e-science* para que os sistemas produzidos atendam às necessidades reais dos membros desta comunidade, ou seja, seus usuários.

O desenvolvimento de tecnologias para *Grids* computacionais deve ser feito visando à integração de recursos e serviços distribuídos de uma forma, ao mesmo tempo, transparente ao usuário e eficiente. Este fator favorece o desenvolvimento orientado a serviços, o que nos remete aos serviços *web,* que seguem a padronização *Web.* O Fórum Global de *Grid* (GGF – *Global Grid Forum*), que é um fórum de pesquisadores que utilizam e desenvolvem aplicações de *Grid,* representa, atualmente, o maior esforço para padronização de *Grids.* Este fórum é composto por grupos de trabalho de várias áreas como serviços de informação, compartilhamento e gerenciamento, dados e modelos, aplicações, segurança, desempenho e arquitetura (LAUSCHNER, 2005).

Um exemplo de especificação GGF é a Arquitetura de Serviços *Open Grid* (OGSA – *Open Grid Service Architecture*), que tem como parte principal o *Grid Service.* A OGSA é uma arquitetura de serviços básicos utilizados na construção de uma infra-estrutura de computação em *Grid* que tem o objetivo de padronizar os principais serviços que são encontrados na maioria das aplicações de *Grid,* especificando interfaces padrões para estes serviços. Esta arquitetura gerou a OGSI (*Open Grid Services Infrastructure*), que descreve as funcionalidades de *Grid Services* que não são encontradas nas especificações de serviços *web* (LAUSCHNER, 2005).

A evolução dos padrões estão se dando com considerável velocidade não só em função das necessidades da comunidade de *e-science*, mas também por causa da evolução dos serviços *Web,* nos quais se baseiam os padrões de computação em grade. Existe uma convergência de padrões de *Grid* e padrões de serviços *Web,* visando fazer com que as ferramentas de *Grid* tenham maior interoperabilidade com os serviços *Web* (LAUSCHNER, 2005).

Para que os sistemas de *Grids* sejam mais utilizados, além de não repetir os erros encontrados na maioria das ferramentas de *Grid* já existentes, os desenvolvedores têm que produzir sistemas padronizados e têm que se preocupar em construir sistemas de qualidade.

Capítulo 3

Workflow para *e-science*

O conceito de *workflow* pode ser definido, em linhas gerais, como um modelo que define o fluxo de processos ou fluxo de tarefas coordenadas e encadeadas usando um plano sistemático. Diz respeito à organização de um conjunto de procedimentos, em que documentos, informações ou tarefas são trocados entre os participantes. A distribuição de tarefas entre os participantes é baseada em regras e requisitos, constantes na definição do *workflow*. A colaboração de todos os participantes é necessária para alcançar um objetivo global de um processo, podendo ser realizada manualmente ou automaticamente (FERNANDES e NOVAIS, 2007).

Um elemento importante para a pesquisa científica é o *workflow*. Mais especificamente, em *e-science*, é necessária a definição de *workflows* científicos, que são *workflows* com alto grau de flexibilidade, permitindo modificações em tempo de execução, que contenham soluções para a ocorrência de incertezas em sua execução e que tratem a ocorrência de exceções (DIGIAMPIETRI, 2007).

Com relação à flexibilidade esperada de *workflows* científicos, temos que sistemas de gerência de *workflow* geralmente interpretam rigidamente a definição de um *workflow*, não permitindo qualquer tipo de desvio durante a execução. No entanto, existem situações reais em que usuários devem poder desviar do fluxo pré-definido por razões, como a falta de informação do valor de um parâmetro ou a indisponibilidade de recursos necessários à execução. Para alcançar uma execução flexível, pode ser usado um mecanismo de tratamento de exceções, voltado para flexibilização, que permite a continuação da execução de uma instância que deveria ser momentaneamente interrompida (VIEIRA, 2005).

São características de *e-science:* o acesso a uma vasta coleção de dados, utilização recursos computacionais em larga escala, utilização de recursos heterogêneos e dinâmicos de múltiplas organizações e *workflows* dinâmicos. Para controlar e facilitar a utilização dos diversos recursos e possibilitar a análise eficiente de grande volume de dados são definidos *workfows* com características específicas para *e-science*. Um *workflow* para *e-science* é um *workflow* voltado para explicitar representações de processos experimentais científicos, geralmente de natureza colaborativa (FERNANDES e NOVAIS, 2007).

Para facilitar o trabalho dos pesquisadores, poupando tempo e possibilitando descobertas cada vez maiores e mais confiáveis, os processos científicos são mapeados para *workflows.* Com a utilização de *workflows,* aplicativos ou serviços estão associados a cada tarefa de experimentos *in silico*, por exemplo, e isso demanda o desenvolvimento e a utilização de mecanismos que permitam a integração entre esses diversos aplicativos e serviços que são utilizados na execução dessas tarefas (MATTOS e MATTOSO, 2007).

Em um ambiente de *e-science* são necessárias ferramentas para projeto e composição de *workflows.* As ferramentas utilizadas devem permitir a visualização do *workflllow* e a visualização de resultados finais e intermediários dos experimentos realizados (FERNANDES e NOVAIS, 2007). Para possibilitar a obtenção de resultados significativos na utilização de *workflows* de bioinformática, por exemplo, são necessários sistemas para gerenciá-los, que auxiliem os pesquisadores na definição, validação, otimização e execução de *workflows* utilizados na execução de experimentos (LEMOS, 2004).

A maioria das tarefas de *e-science* são processadas em recursos distribuídos e heterogêneos, para alcançar um objetivo específico. A utilização de recursos distribuídos se dá devido à necessidade de processamento de grandes volumes de informação sem perda de desempenho, além de permitir que resultados oriundos de diversos grupos de pesquisa ou de diversas unidades de uma mesma organização, separadas geograficamente, possam ser integrados (FERNANDES e NOVAIS, 2007).

A utilização de *grids*[4] computacionais apresenta também certa facilidade de construção de aplicações dinâmicas (FERNANDES e NOVAIS, 2007). Nesses ambientes distribuídos podem ser definidos e executados os *workflows* para *grid*. Esses *workflows* localizam os recursos disponíveis no ambiente de *grid,* controlam a distribuição de atividades entre esses recursos, além de controlar a execução das tarefas mantendo sua ordem e considerando as dependências entre elas. O acesso a dados e a execução de *workflows,* de forma eficiente, em um ambiente de *grid* computacional, utilizando a infra-estrutura de *grid* disponível atualmente, é uma tarefa complexa.

Um *workflow* definido para ser executado em um ambiente de *grid* é um *grid workflow*, que pode ser definido como uma coleção de tarefas, a serem processadas utilizando-se recursos distribuídos, em uma ordem predefinida. Um *grid workflow* tende a ser mais longo que um *workflow* comum e deve suportar grandes fluxos de dados. Os

[4] Um *Grid* Computacional trata do compartilhamento de recursos relacionados e soluções de problemas, em organizações virtuais distribuídas geograficamente (LAUSCHNER, 2005).

recursos utilizados por este tipo de *workflow* são heterogêneos e distribuídos por diversos domínios administrativos diferentes, além disso, a disponibilidade para utilização dos recursos pode variar em tempo real, durante a execução do *workflow* (LEMOS, 2004). Alguns problemas como o desenvolvimento de formas de execução dinâmica de *grid workflows*, a dificuldade para definir a localização de dados intermediários e a dificuldade de obtenção de informações sobre os recursos a serem utilizados, desafiam o desenvolvimento e execução de *grid workflows* (MATTOSO, 2007).

Em um *workflow* se tem um conjunto de dados de entrada a serem processados. Os dados devem ser processados da maneira mais eficiente possível, pois o desempenho do sistema é um fator tão importante quanto a correta execução da tarefa. Em um ambiente de *grid* existem outros sistemas, de terceiros, concorrentes ao que se quer executar, com tempos de execução diferentes, volumes de dados variados e ambientes computacionais distintos. Estes outros processos devem ser levados em conta para que a execução de um *workflow* que envolva vários processos, de diversas origens, não seja prejudicada. Alguns programas têm dependência em relação a outros e isso também pode gerar dificuldades de decisão no momento da execução de um *workflow* (MATTOSO, 2007).

Para contornar esses problemas na execução de *workflow* em ambiente distribuído, é necessária a integração dos processos envolvidos na execução do *workflow*. Para gerenciar os recursos de um *workflow* podem ser utilizadas tecnologias de serviços *Web* como linguagens de definição de *workflows* de serviços, planos de execução ou rastreamento de execução (MATTOSO, 2007). Para executar um *grid workflow* é necessária a utilização de ferramentas combinadas, utilização de recursos compatíveis com o ambiente do usuário e capacidade de transferência de dados em larga escala (LEMOS, 2004).

3.1 *Workflow*: origem e conceitos

Sistemas de gerência de *workflow* são bastante antigos, mas tiveram a sua popularidade acentuada apenas nas últimas décadas, em decorrência principalmente do seu uso na área de negócios nas empresas. Historicamente, sistemas de gerência de *workflow* têm sua origem na automação do trabalho. Eles eram, no entanto, monolíticos de modo que as políticas de negócio e acesso às informações estavam rigidamente definidas nas aplicações. Além disso, estes sistemas quase sempre se encontravam isolados, sem ligação alguma com outros sistemas de *workflow* ou aplicações genéricas (VIEIRA, 2005).

A primeira geração dos sistemas de *workflow* compreendeu aplicações monolíticas de uma área de domínio particular. A segunda geração já dividiu os sistemas em componentes diversos, mas eles ainda eram fortemente dependentes das aplicações. A terceira geração apresentou máquinas de *workflow* genéricas que forneciam uma infra-estrutura robusta para *workflows* orientados à produção. Nesta geração, a descrição dos *workflows* era construída através de ferramentas gráficas e, posteriormente, era interpretada pelas máquinas de *workflow*, responsáveis por sua execução. Uma quarta geração seria a que se está presenciando nos últimos anos, a de sistemas de *workflows* que oferecem uma gama de serviços (VIEIRA, 2005).

Com a evolução natural dos sistemas, tornou-se necessário separar processos de negócio e componentes de *workflow* das aplicações existentes, visando aumentar a flexibilidade e o poder de manutenção destes sistemas. Além disso, os sistemas passaram de "orientados a dados" para "orientados a processos". A descentralização das organizações e da tomada de decisões, a necessidade por informação detalhada sobre as atividades diárias, assim como a ênfase na arquitetura cliente/servidor e mais recentemente arquiteturas distribuídas, a relevância dos sistemas federados e a crescente disponibilidade de tecnologia para processamento distribuído indicaram o fim do processamento centralizado e monolítico (VIEIRA, 2005).

Os sistemas de *workflow* se inserem na área de *software* de apoio ao trabalho colaborativo, cujo objetivo é o de propiciar suporte computadorizado a grupos de pessoas que trabalham em conjunto para atingir um objetivo comum, auxiliando as organizações na especificação, execução, monitoramento e coordenação do fluxo de trabalho em um ambiente de trabalho distribuído. Estes sistemas podem ser utilizados em diversas áreas, como a de engenharia, a científica e a organizacional (BARTHELMESS, 1996).

Atualmente, a tecnologia de *workflow* não se limita à gerência de documentos em organizações. Os conceitos e paradigmas de trabalho em grupo preconizados pelas pesquisas em Trabalho Colaborativo Suportado por Computadores - CSCW (*Computer Supported Cooperative Work*) e *groupware* a partir da década de 80 influenciaram a declaração destes sistemas como ferramentas para a coordenação do trabalho de equipes e impulsionaram seu desenvolvimento. Sob esta visão, a tecnologia de *workflow* visava unir as chamadas "ilhas de informação e trabalho" individuais e personalizadas de cada parte de uma organização, buscando sua integração por meio do roteamento do trabalho entre elas (ARAUJO e BORGES, 2001).

Um *workflow* atual representa, portanto, um conjunto de atividades a serem executadas, suas relações de dependência, dados de entrada e dados de saída (DIGIAMPIETRI; MEDEIROS e SETUBAL, 2005). A figura 3 ilustra a representação de um *workflow*.

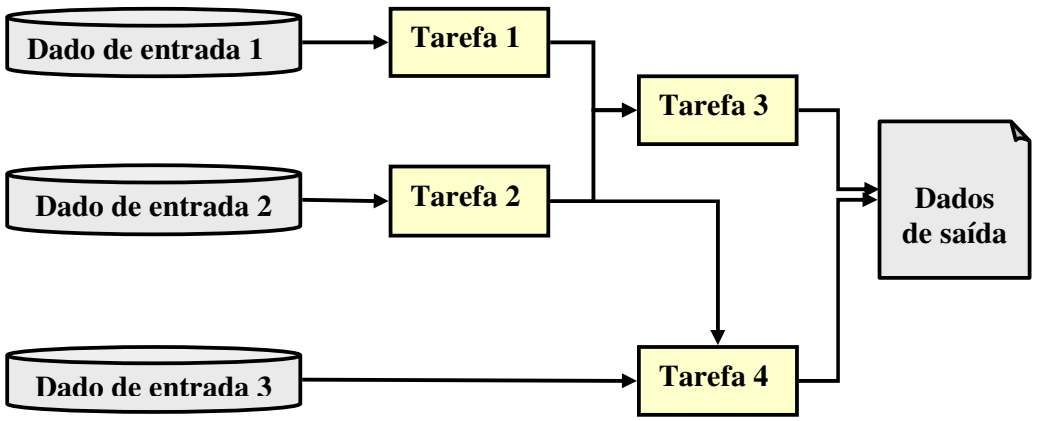

Figura 3 – Representação de um *workflow* (DIGIAMPIETRI; MEDEIROS e SETUBAL, 2005)

3.2 *Workflow* para negócios

Num contexto de grandes e rápidas mudanças, com a exigência dos clientes por níveis cada vez mais altos de qualidade e personalização do atendimento a suas necessidades, as empresas e organizações estão adotando novos modelos de gestão e de organização social do trabalho. As organizações tomam também consciência da importância cada vez maior da necessidade de aprendizagem e da gestão do seu conhecimento. Como resposta a estas exigências, é requerida à organização uma constante reengenharia e melhoria de cada processo de negócio. A adoção dos novos modelos de trabalho é acompanhada pela adoção de tecnologias e sistemas de informação que permitam às organizações lidarem com a informação necessária, com qualidade e precisão. (LOUSÃ; SARMENTO e MACHADO, 2007; ARAUJO e BORGES, 2001).

Neste contexto, a estrutura de trabalho vem ressaltando a necessidade de enfocar o trabalho em equipe, a formação e reuso do conhecimento comum dentro da organização e a melhoria contínua dos processos de produção. A tecnologia de *workflow* está associada a estas modificações por oferecer um conjunto de soluções para atender à demanda das organizações por maiores índices de qualidade e eficiência no gerenciamento de seus processos de negócio (ARAUJO e BORGES, 2001).

Desde suas origens, sistemas de *workflow* têm encontrado grande demanda no ambiente de negócios. Por isso, esta tecnologia evoluiu muito no sentido de apoiar as novas necessidades de relacionamento e execução de atividades em organizações inseridas em um mundo globalizado, se tornando uma peça chave para soluções de apoio automatizado a atividades e formas de negócio em larga escala (ARAUJO e BORGES, 2001).

Workflows podem ser associados a processos de organizações. Várias tarefas são executadas, em ordem definida, respeitando as dependências entre elas, quando essas dependências existem, e podendo ser executadas de forma paralela, caso sejam independentes. A principal função de um sistema de gerenciamento de *workflow* é a distribuição de tarefas a executores diferentes, gerenciando a sincronização entre essas tarefas e respeitando a interdependência que possa existir entre elas (BARTHELMESS, 1996).

A *Workflow Management Coalition* (*WfMC*), que é uma organização para definição de padrões de *workflows,* em 1996, definiu *workflow* como a automatização de um processo de negócio ou de parte dele, englobando documentos, informações ou tarefas que são passadas entre os participantes do processo, de acordo com regras de negócios pré-definidas. Um processo de negócio é uma instância de uma tarefa bem definida que é freqüentemente repetida como parte de uma regra de negócio (TAYLOR *et al.*, 2006).

A *WfMC* oferece um grande conjunto de modelos de referência, documentos e padrões. Esses padrões são, em geral, voltados para o conceito de *workflows* de negócios e, portanto, não podem ser integralmente utilizados também para uso em *workflows* científicos por existirem características que são exclusivas dos *workflows* científicos. Apesar dessas diferenças, alguns conceitos podem ser utilizados tanto em *workflows* de negócios como em *workflows* científicos (TAYLOR *et al.*, 2006).

Leyman e Roller caracterizaram quatro tipos básicos de *workflows* de negócios. A maioria deles tem seus correspondentes em *workflows* aplicáveis em ciências e engenharia. São eles (TAYLOR *et al.*, 2006):

- *Workflow* colaborativo (*Collaborative workflow*). É o tipo mais utilizado em negócios por permitir um grande grupo de participantes em um processo, não deixando de englobar os processo individuais. Em *e-science*, este *workflow* pode ser utilizado no gerenciamento de dados produzidos, e para grandes experimentos executados de forma distribuída, com a participação de diversos colaboradores.

- *Workflow ad-hoc*. Este *workflow* pode ser utilizado para tarefas menos formais, com definição menos rígida como, por exemplo, a notificação de alguma modificação de procedimento de negócio. Este tipo de tarefa não depende da interação entre muitas partes e é, portanto, independente. Em *e-science*, as atividades de notificação são muito constantes, como, por exemplo, a notificação de entrada de dados em um sensor, que dispara algum tipo de processamento.
- *Workflow* administrativo (*Administrative workflow*). Este *workflow* se refere às atividades relacionadas à administração empresarial, como, por exemplo, a gerência de bancos de dados e a manutenção de agendas. Um exemplo de aplicação deste *workflow* em *e-science* é a rotina de tarefas usada para gerenciar os dados vindos de instrumentos de monitoração ou sensores.
- *Workflow* de produção (*Production workflow*). Está relacionado a processos de negócio que definem as principais atividades de uma organização. Apresenta tarefas que são repetidas frequentemente, podendo apresentar processos concorrentes. Este *workflow* é utilizado, em *e-science*, para análise de dados e simulações diárias, como, por exemplo, em análise de dados meteorológicos.

Existem, porém, *workflows* de negócios que são muito diferentes de seus correspondentes aplicados às ciências. Uma característica peculiar de um *workflow* de negócios é a necessidade de segurança e integridade de uma seqüência de ações. Esta característica, na maioria das vezes, não é comum aos *workflows* científicos. Esta necessidade de execução garantida de uma seqüência de tarefas, vêm do fato de que os *workflows* de negócio são, muitas vezes, baseados em transações. Transações podem ser definidas como um conjunto de tarefas que devem ser executadas em sua totalidade ou então não serão executadas. Alguns *workflows* não podem seguir o modelo de transações. Um *workflow* muito longo, por exemplo, pode conter em suas especificação, *subworkfows* que deveriam ser executados como transações, porém, nem sempre isso é possível devido à natureza das tarefas executadas ao longo do *subworkflow* (TAYLOR *et al.*, 2006).

A utilização da tecnologia de *workflow* nas organizações está ligada à busca por maior eficiência, à racionalização de custos e ao maior controle sobre suas operações. A busca por maior eficiência está mais ligada à forma como os processos são definidos e executados, isto é, a reengenharia de processos e não tem muita ligação com a automação do controle de execução dos mesmos. A racionalização de custos está relacionada à

diminuição da circulação de documentos de papel. O maior controle sobre as operações vem da melhoria na gerência da execução de processos e um maior controle sobre o trabalho das pessoas (ARAUJO e BORGES, 2001).

A implantação de uma solução ligada à tecnologia de *workflow* implica em mudanças de diversos aspectos na organização, o que é um processo complexo. Por isso, a introdução de uma tecnologia de *workflow* deve ser cuidadosamente planejada e acompanhada passo a passo para ser adaptada às circunstâncias geradas pela própria introdução da tecnologia. Uma forma de amenizar essa transição é aplicar a tecnologia gradativamente. Apesar da tecnologia de *workflow* ser de difícil implantação, ela pode oferecer diversas vantagens às organizações, tais como a redução dos custos de operação, melhor controle sobre as operações, monitoramento do trabalho realizado, melhoria do atendimento ao cliente e menor circulação de documentos em papel (ARAUJO e BORGES, 2001).

O melhor benefício proporcionado pela implantação de uma solução de *workflow* é a possibilidade de se ter um maior controle sobre os processos realizados pela empresa. Esse controle facilita o trabalho de quem gerencia processos, de quem necessita de informações sobre o andamento de determinado processo e possibilita maior qualidade na prestação de serviços. Ao automatizar a operação, a conseqüência direta é um melhor controle. Normalmente, o objetivo inicial da automação são os processos críticos de uma empresa, mas o ideal é a automação de toda a cadeia de processos da organização (ARAUJO e BORGES, 2001).

3.3 *Workflow* científico

A noção de integração da informação deve ser ampliada para incluir o paradigma emergente de *workflows* científicos. No domínio científico, a integração da informação frequentemente significa um processo fim-a-fim, consistindo na aquisição de dados, sua transformação, análise, visualização, entre outros passos. A integração de informação científica requer um novo paradigma que combine uma abordagem orientada a dados com uma abordagem de modelagem de *workflow* orientada a processos apropriada. Para satisfazer essa necessidade é necessário (i) olhar os dados científicos e as questões de gerenciamento de *workflows* com olhos de cientistas e (ii) resolver questões de pesquisas multidisciplinares decorrentes dessa nova perspectiva (LUDÄSCHER, 2007).

Um *workflow* científico (*cientific workflow*) é a especificação de um processo que descreve um experimento científico. Enquanto, em sua maioria, os *workflows* de negócios (*business workflow*) são especificados primeiro e só depois são executados, um *workflow* científico normalmente é especificado durante a condução do experimento, focado na documentação do processo. Outras características singulares deste tipo de *workflow* incluem: (i) ênfase em processos centrados em dados; (ii) um alto grau de flexibilidade com relação à especificação e à adaptação; (iii) suporte a incertezas na especificação e no tempo de execução, como por exemplo, grande quantidade de exceções durante a execução; (iv) possibilidade de modificação do *workflow* durante sua execução (PASTORELLO JR; MEDEIROS e SANTANCHÈ, 2007).

Os *workflows* científicos podem ser considerados como cruciais na interação dos cientistas com a infra-estrutura computacional aplicada à pesquisa. Eles facilitam as atividades de *e-science* permitindo que os cientistas modelem, executem, verifiquem, reconfigurem e re-executem as análises de dados científicos. Este tipo de *workflow* também pode ser visto como um novo paradigma de modelagem que integra o controle de dados e a computação orientada a processos (LUDÄSCHER, 2007).

As características dinâmicas da definição de *workflows* científicos geram dificuldades no momento da especificação do *workflow*, e para resolver este problema torna-se necessário o uso de metodologias específicas para a especificação de *workflows* científicos. Para gerenciar *workflows* científicos pode ser usado um *framework* especialmente projetado para este fim (PASTORELLO JR; MEDEIROS e SANTANCHÈ, 2007).

Na maioria dos *frameworks* para gerência de *workflows* os passos do *workflow* são modelados como componentes que estão ligados pelo fluxo de dados que circula entre eles. Os componentes são como processos independentes que se comunicam por mensagens enviadas por canais unidirecionais. Esse fluxo de dados orientado a processos provê aos cientistas que modelam seus *workflows* um modo simples de modelar suas atividades. Esse modelo é apropriado para representar o *workflow* em nível conceitual expondo tarefas e paralelismos (LUDÄSCHER, 2007). Para que os cientistas analisem a execução dos *workflows*, entendam os resultados obtidos a partir dessa execução e possam re-executar tarefas, o sistema de gerência de *workflow* deve apresentar a habilidade de registrar dados de proveniência, de preferência, de forma automática. As informações obtidas a partir dos dados de proveniência são úteis às atividades científicas relacionadas à análise dos resultados de experimentos.

Da perspectiva de um cientista, a integração das informações científicas obtidas a partir de diferentes experimentos é necessária para a formulação de conclusões e, envolve a definição de *workfllows* e experimentos, sua execução e gerência (permitindo reuso, armazenamento e proveniência). Para realizar esta integração, bancos de dados podem ser usados para armazenar os dados científicos produzidos pelas pesquisas, mas uma síntese de processos e integração de informações via *workflows* científicos é mais eficiente para satisfazer as necessidades de *e-science* (LUDÄSCHER, 2007). A criação de novos modelos, a utilização de modelos já existentes, a manipulação de dados novos e antigos, a realização de análises sobre os dados e visualização dos resultados podem ser obtidos a partir da definição e execução de *workflows* científicos (PENNINGTON, 2007).

Os *workflows* científicos estão sendo aplicados com sucesso em diversas áreas da ciência, possibilitando aos cientistas a realização de análises complexas de problemas que envolvem grande volume de dados e necessitam de busca de ferramentas. No entanto, os desafios técnicos dos *workflows* científicos são complexos, e esse fator faz com que a variedade de tarefas suportadas por esses *workflows* seja muito limitada, se comparada ao conjunto completo de tarefas necessárias à realização dos experimentos pelos cientistas. O trabalho da ciência é muito mais abrangente que a visão atual de *workflows* pode sugerir (PENNINGTON, 2007).

A modelagem de *workflows* científicos ainda é muito difícil, pois existe um maior domínio geral na modelagem de *workflows* para negócios (*business workflows*), visto que as características deste segundo tipo de *workflow* são bem diferentes das características do primeiro. A diferença mais notável entre os *workflows* científicos e de negócios é a escala. Quando comparados aos *workflows* científicos, os *workflow*s para negócios normalmente são bem menores. Os *workflows* científicos podem envolver centenas de instâncias de serviços, que terão que ser modeladas. Além disso, os *workflows* científicos, na maioria das vezes, vão executar centenas de invocações de serviços básicos e, consequentemente, vão enviar centenas de mensagens, que serão trocadas entre os serviços. Os *business workflows*, normalmente, operam em pequena escala (TAYLOR *et al.*, 2006).

Outra diferença diz respeito às necessidades de modelagem de execuções paralelas nos *workflows* científicos. Os *workflows* científicos aplicam modelos computacionais complexos que geram grandes quantidades de dados e, só então, estes dados são analisados. Portanto, estes *workflows* têm uma grande quantidade de *subworkflows* independentes que têm que ser executados de forma concorrente. *Workflows*

de negócio normalmente não necessitam de execução paralela massiva de diversos *subworkflows* paralelos, pois, em geral, são menores (TAYLOR *et al.*, 2006).

3.4 BPEL: uma linguagem para *workflow*

A *Business Process Workflow Language for Web Services* (BPEL4WS) (OASIS, 2007), quando foi criada, em 2002, veio para substituir duas linguagens para *workflow* criadas anteriormente pela IBM e pela Microsoft. A *Web Services Flow Language* (WSFL) (WSFL, 2007), que foi criada pela IBM, disponibilizava uma visão orientada a grafos para definir um *workfllow*. A *XLANG* (XLANG, 2007), criada pela Microsoft apresentava uma visão mais voltada para a estrutura orientada a blocos (TAYLOR *et al.*, 2006).

A BPEL4WS está inserida no contexto de serviço *web* como seu próprio nome sugere. Todas as interações internas a um *workflow* ocorrem através de interfaces de serviços *web* definidas em WSDL. Dessa forma, os processos podem apresentar dois comportamentos: (1) o processo interage com o serviço *web* através da interface dos serviços *web* descrita em WSDL e (2) o processo define a si mesmo como um serviço *web* descrevendo sua interface por meio de WSDL (LEYMANN; ROLLER e THATTE, 2007).

A BPEL fundiu as abordagens existentes e adicionou extensões de suporte ao tratamento de erros com a estrutura *try/catch*. A versão inicial de BPEL (versão 1.0) foi seguida, em 2003, pela versão 1.1 que clareou e melhorou diversas partes de BPEL 1.0. No mesmo ano, BPEL foi submetida à *Organization for the Advancement of Structured Information Standards* (OASIS), que é uma organização que trabalha para definir padrões para a informação estruturada, e desde 2004 vem sendo padronizada como WS-BPEL 2.0. A OASIS WS-BPEL 2.0 é uma versão revisada da antiga linguagem e, por isso, passou a não ser mais totalmente compatível com as versões 1.x (TAYLOR *et al.*, 2006).

Como a BPEL4WS foi desenvolvida para trabalhar com serviços *web*, cada BPEL *workflow* pode ser considerado um serviço *web*. Isso faz BPEL se ajustar perfeitamente ao *middleware* de serviços *web* e torna fácil a composição de *workflows* hierárquicos. Um BPEL *workflow* é um serviço *web* que pode ser usado dentro de um outro BPEL *workflow* que, por sua vez, também pode ser usado dentro de um outro BPEL *workflow* e assim por diante. Essa estrutura permite a composição de um *workflow* a partir de vários outros (TAYLOR *et al.*, 2006).

A especificação de BPEL oferece um vasto conjunto de ferramentas para manipulação de mensagens XML, extração e combinação de partes de mensagens XML,

descrição e validação do conteúdo dessas mensagens e roteamento de mensagens para serviços *web* (TAYLOR *et al.*, 2006). BPEL4WS define processos de negócios usando uma linguagem baseada em XML e não se concentra em representações gráficas dos processos nem especifica uma metodologia de definição particular para os processos (LEYMANN; ROLLER e THATTE, 2007).

A BPEL4WS possibilita a definição de um conjunto de conceitos de composição de serviços que podem ser utilizados nas definições abstrata e executável de um processo de negócio, que define o comportamento de uma entidade autônoma. BPEL permite a descrição de um "BPEL abstrato" que é um modelo que destaca os comportamentos mais importantes do *workflow*, sem especificar todos os detalhes. A intenção é permitir uma definição a partir das características públicas visíveis do *workflow*, escondendo detalhes que podem ser variáveis em implementações diferentes do mesmo *workflow*. Isso funciona como uma interface em linguagens de programação. O "BPEL abstrato" será então implementado por um BPEL *workflow* que irá conter todos os detalhes e será chamado "BPEL executável" (TAYLOR *et al.*, 2006).

A linguagem tem um forte conjunto de estruturas de controle (laços, condições etc.) e um bom suporte para capturar e manipular exceções (falhas) e reversão de mudanças pelo uso de compensações. Compensações são importantes, em particular, para *workflows* com execuções longas, que necessitam de operações de "*undo*" (desfazer) em casos de erros irrecuperáveis em serviços que estão sendo usados pelo *workflow*, em que consistências globais têm que ser restauradas antes que o *workflow* seja finalizado (TAYLOR *et al.*, 2006).

A estrutura completa de um BPEL *workflow* está representada na figura 4.

```
<process name="BpelProcessName" targetNamespace="..."
    xmlns="http://schemas.xmlsoap.org/ws/2004/03/business-process/">
  <partnerLinks>
     <partnerLink name="partnerA" partnerLinkType="wsdl:parnterALinkType"
       myRole="myRoleInRelationToPartner"/>
     ...
  </partnerLinks>
  <variables>
    <variable name="varA" messageType="wsdl:MessageA"/>
    ...
  </variables>
  <!-- this is executable part of workflow -->
  <sequence>
     <receive partnerLink="partnerA" portType="wsdl:parnterALinkType"
       operation="doSomething" variable="varA" />
     <assign>
       <copy>
         <from>$varA.someParameter</from>
         <to>$varB.anotherInfo</to>
       </copy>
     </assign>
     <invoke partnerLink="partnerB" portType="pb:anoptherPartnerPT"
       operation="doSomethingElse" inputVariable="varB"
       outputVariable="varC" />
     .... <!-- here something more happens -->
     <reply partnerLink="partnerA" portType="wsdl:parnterALinkType"
        operation="doSomething" variable="results"/>
  </sequence>
</process>
```

Figura 4 - Estrutura de um BPEL *workflow* (TAYLOR *et al.*, 2006)

A definição de um BPEL *workflow* está contida em um elemento `<process>`. Este elemento é composto por diversos outros elementos, como `<partnerLinks>` e `<variables>`, e uma atividade que representa um ponto de entrada para o *workflow* (normalmente isto é representado pelo elemento `<sequence>`) (TAYLOR *et al.*, 2006).

BPEL provê um conjunto de construções simples para enviar e receber mensagens. Tipicamente, um BPEL *workflow*, irá começar com uma atividade de recebimento de mensagem (`<receive>`) e irá terminar com uma atividade resposta de mensagem (`<reply>`), que irá responder à mensagem recebida inicialmente. É fácil enviar uma mensagem para outro serviço *web* (que são chamados *partners* em BPEL) usando a atividade de invocar (`<invoke>`) outros serviços. Existem duas versões de invocação, sendo uma versão em que apenas uma variável de entrada está presente, e a outra em que se está respondendo a uma requisição e uma variável de entrada e uma de saída estão presentes (TAYLOR *et al.*, 2006).

A linguagem oferece funções limitadas para manipulação de dados. Essas funções são suficientes apenas para a manipulação simples dos dados mais relevantes ao processo e para o controle do fluxo dos dados do processo (TAYLOR *et al.*, 2006).

Todas as mensagens em BPEL estão contidas em variáveis. As variáveis são passadas entre atividades BPEL. Para copiar e modificar o conteúdo de variáveis pode ser usada a atividade de assinalar (`<assign>`). É suportada a linguagem XPath para selecionar e modificar o conteúdo XML (TAYLOR *et al.*, 2006).

A BPEL4WS apresenta ainda um mecanismo de identificação para instâncias de processos, que permite a definição de identificadores de instâncias no nível do envio de mensagens da aplicação. Além do controle de identificadores, BPEL4WS também oferece suporte à criação e destruição de instâncias de processos por meio de um mecanismo básico de controle de ciclo de vida (TAYLOR *et al.*, 2006).

A linguagem apresenta o recurso de definir *workflows* baseados em grafos. É fácil iniciar diversas atividades em paralelo com `<flow>` e BPEL permite definir dependências, de forma gráfica, entre as atividades. Cada atividade (um nó no grafo) pode ter diversos *links* de chegada e de saída. Para uma atividade iniciar sua execução, todos os *links* de chegada devem estar habilitados. Quando uma atividade é finalizada, todos os seus *links* de saída ficarão habilitados. Sendo estes *links* de entrada de outras atividades, estas atividades poderão ser iniciadas. Este recurso permite que sejam construídos diversos tipos de grafos em BPEL e a parte mais interessante é que BPEL permite ao programador misturar abordagens estruturadas e gráficas em um mesmo *workflow* (TAYLOR *et al.*, 2006).

A BPEL não tem *loop* paralelo. Este fator é particularmente importante quando se trata de uma aplicação científica. Se o número de iterações é constante, é possível usar `<flow>` para iniciar múltiplas atividades em paralelo, mas essa abordagem não funciona se o número de iterações depende de um dado de entrada do *workflow*, por exemplo. Um *loop* paralelo pode ser simulado com invocações não bloqueantes de um serviço *web* (que será um BPEL *sub-workflow*). Mas esse tipo de invocação é difícil de definir e, em geral, estabelecer canais de comunicação entre o *sub-workflow* e o *workflow* principal, assim como detectar se todos os *sub-workflows* foram finalizados com sucesso, pode ser uma tarefa difícil (TAYLOR *et al.*, 2006).

Existem diversas ferramentas que podem ser usadas para definir BPEL *workflows*. Elas podem variar de editores de XML simples ou mais sofisticados até ferramentas gráficas com uma interface para compor *workflows* através da conexão de serviços *web* por meio de grafos, escondendo do usuário o texto XML dos processos de BPEL e gerando o

XML automaticamente quando necessário. Como as ferramentas gráficas operam em um alto nível de abstração elas, normalmente, suportam apenas um subconjunto das funcionalidades de BPEL. Essas funcionalidades costumam estar relacionadas a um determinado grupo de usuários ao qual a ferramenta se aplica (TAYLOR *et al.*, 2006).

A BPEL vem se tornando o principal padrão para *workflows* baseados em serviços *web* e pode ser integrada à arquitetura *web* e a portais. Uma das maiores limitações de BPEL são um suporte pobre para a execução de um grande número de *sub-workflows* paralelos. Apesar das limitações, esta linguagem é uma escolha viável para *workflows* de *grid*, mas são necessários recursos adicionais para sejam atendidas todas as necessidades da execução deste tipo de *workflow*. BPEL suporta extensões como BPELJ, que permite interações entre BPEL *workflows* e componentes Java, e é possível que algumas das extensões existentes se tornem padrões para BPEL *workflows* científicos de *grids* (TAYLOR *et al.*, 2006).

Por enquanto, a BPEL não é a linguagem mais utilizada para *workflow* em projetos científicos, apesar de apresentar características muito ligadas a *workflows* científicos como, por exemplo, recursos para o tratamento de exceções e facilidades para representação de *workflows* complexos, além de ser uma boa linguagem para compartilhamento de *workflows* entre diferentes projetos, o que facilita atividades de *e-science*, fortalecendo a idéia de que esta linguagem pode ser usada no apoio à pesquisa científica. A BPEL pode ser vista como uma ferramenta que permite que um *workflow* importe e exporte outros BPEL *workflows* em projetos científicos (TAYLOR *et al.*, 2006).

Para que a BPEL possa ser largamente utilizada na modelagem de *workflows* científicos, é necessária a extensão da linguagem com adição de abstrações de mais alto nível que as originalmente oferecidas por BPEL. Para que essas extensões sejam possíveis, é necessária a utilização de ferramentas de modelagem que ofereçam este tipo de suporte. Existem, atualmente, muitos esforços no sentido de desenvolver ferramentas que possibilitem extensões à linguagem e facilidades ao desenvolvimento de *workflows* científicos, o que será visto no próximo capítulo.

Capítulo 4

Ferramentas para *e-science*

Como a pesquisa científica envolve vários programas de análise e bancos de dados, que estão em constante crescimento e atualização, é imensa a possibilidade de análises que podem ser feitas. Desta forma, para se ter produtividade, eficiência e qualidade nas análises, torna-se necessária à definição de uma linguagem de *workflow* apropriada com suporte de um sistema de gerência de *workflows* (SGW) (LEMOS, 2004).

Um SGW oferece automatização dos procedimentos de um negócio ou experimento científico através do gerenciamento de uma seqüência de atividades, e da invocação dos recursos humanos e tecnológicos apropriados, associados aos diversos passos da atividade.

Segundo LEMOS (2004), temos uma lista de requisitos que um sistema de gerência de *workflows* deve atender:

- O sistema deve incluir os processos, dados e recursos normalmente usados e oferecer mecanismos de extensibilidade para acomodar novos processos, dados e recursos;
- O sistema deve permitir ao usuário a definição e redefinição de *workflows*, permitindo a definição de propriedades dos processos, dados e recursos facilitando a escolha do pesquisador de acordo com sua necessidade;
- O sistema deve oferecer ferramentas de validação do *workflow* definidas pelo pesquisador. Devem ser verificadas as entradas e saídas definidas pelo usuário para cada processo do *workflow* verificando sua coerência. Caso sejam necessários, devem ser definidos processos que façam conversão de formato de dados e processos que verifiquem se os resultados gerados pelos processos são esperados ou não. Em caso de resultados não esperados, pode ser necessário o tratamento de alguma exceção;
- O sistema deve ser capaz de otimizar e executar o *workflow* definido pelo pesquisador, de acordo com a arquitetura que está sendo utilizada. A execução do *workflow* pode ser monitorada pelo pesquisador e a intervenção do mesmo sobre esta execução deve ser possível, no caso de necessidade de avaliação de resultados intermediários. Ao avaliar os resultados intermediários, o pesquisador pode decidir continuar ou não a execução e

também pode fazer alguma modificação na definição das próximas atividades do *workflow*;

- O sistema deve oferecer agendamento da execução do *workflow*;
- O sistema deve armazenar metadados sobre o *workflow*. O sistema deve ser capaz de gerar estes metadados automaticamente e oferecer mecanismos para o usuário consultá-los e atualizá-los. Os metadados poderão ajudar os pesquisadores a definir novos *workflows*, usar resultados gerados em execuções anteriores de *workflows* como entrada de novas execuções de *workflows* e minerar os resultados produzidos por execuções de *workflows* para descobrir informações relevantes.

Além da gerência de *workflows* é necessário ter o suporte de um sistema de gerência de dados científicos. Este sistema deve ter a capacidade de armazenar, acessar e manipular dados e metadados. Além disso, o sistema deve oferecer ferramentas que permitam a criação de consultas sobre os dados armazenados e deve permitir a visualização dos resultados de forma amigável, ajudando o pesquisador a analisar os dados. E, ainda, o sistema de gerenciamento de dados deve, sobretudo, oferecer mecanismos de segurança para o acesso aos dados, não permitindo o acesso por usuários não autorizados (LEMOS, 2004).

Atualmente existem alguns sistemas que tratam de gerência de *workflows* e exploração de dados científicos, sendo possível destacar o MyGrid (Stevens, 2003 *in* LEMOS, 2004), o Taverna (TAVERNA, 2007), o VisTrails (VISTRAIL, 2007) e o Kepler (KEPLER, 2007).

4.1 MyGrid

"O MyGrid é um projeto de desenvolvimento de uma camada *open-source* em alto nível, que pode ser utilizada por diversos laboratórios para a execução de aplicativos e *workflows* em um ambiente de *grid*" (Foster, 2001; IBM Corporation, 2004 *in* LEMOS, 2004).

O projeto MyGrid é uma colaboração entre diversas universidades do Reino Unido e empresas multinacionais. É uma ferramenta que tem como objetivos principais dar suporte à utilização de recursos em um *grid* computacional e contornar dificuldades de gerenciamento de escalonamento de tarefas no ambiente de *grid* (CARDOSO; MATTOSO e MATTOS, 2007). O MyGrid é também uma plataforma de execução de aplicações paralelas leves em *grids*. Aplicações leves, também chamadas aplicações *bag-of-tasks*, são

definidas como aplicações compostas de um conjunto de tarefas independentes que não necessitam de comunicação durante sua execução (ANDRADE, 2002).

O MyGrid tem código aberto implementado em Java e *scripts shell.* É multiplataforma, podendo ser utilizado tanto em ambientes *Linux* como *Windows*. O *grid* de um determinado usuário do MyGrid é composto por todas as máquinas às quais ele tem permissão de acesso. Isto é extremamente interessante para aplicações do tipo *bag-of-tasks*, devido ao fraco acoplamento das tarefas (CARDOSO; MATTOSO e MATTOS, 2007).

Uma ontologia[5], definida em DAML-OIL [Horrocks, 2002; Wroe, 2003; W3C, 2001a, b], oferece um vocabulário controlado de conceitos que descrevem os serviços e *workflows*, tipos de entrada permitidas, tipos de saídas produzidas e recursos utilizados por estes serviços. Esta ontologia é utilizada para classificar e recuperar serviços e *workflows* de acordo com a semântica dos dados de entrada e saída e, de acordo com as tarefas que cada serviço e *workflow* fazem (LEMOS, 2004).

Não é necessária a instalação de nenhuma infra-estrutura por parte dos administradores. As máquinas que compõem o *grid* são vistas pelo usuário através de uma mesma abstração, chamada de *Grid Machine Abstraction*. Existem implementações desta abstração para situações diversas, como, por exemplo, acesso através de uma agente Java ou através de *scripts shell*. Sob essa mesma abstração, é possível ao MyGrid fornecer acesso a recursos controlados por outros *middlewares* (ANDRADE, 2002).

O aspecto operacional do MyGrid é resolvido movendo o poder de instalação e configuração do *middleware* do administrador do sistema para o usuário. A partir do acesso aos recursos, ele mesmo pode configurar sua plataforma global, sem a necessidade da intervenção do administrador. Essa relativa independência do usuário torna o sistema mais flexível e acessível a usuários que não têm uma infra-estrutura de computação em *grid* disponível, mas desejam utilizar recursos disponíveis aos quais têm acesso (ANDRADE, 2002).

O MyGrid não se propõe a resolver o aspecto gerencial. É responsabilidade do usuário, conseguir acesso aos recursos que ele deseja utilizar. Contudo, uma vez que os mecanismos necessários para a utilização de um recurso são mínimos, é mais fácil para os usuários obter acesso e utilizar um maior número de recursos. Por exemplo, um usuário

5 Uma ontologia é uma especificação de conceitos (Gruber, 1993; Guarino, 1998; Stoffel; 1997 *in* LEMOS, 2004), ou seja, uma descrição concisa e não ambígua de quem são as entidades relevantes no domínio da aplicação e como elas se relacionam. As entidades podem ser objetos, processos, funções, predicados e outros tipos dependentes da aplicação" (Schulze-Kremer, 1998 *in* LEMOS, 2004).

pode obter acesso a uma rede de estações de trabalho apenas conseguindo um *login* em um laboratório de uma universidade. Apesar dessa facilidade, entretanto, é importante notar que a maioria dos usuários não possui acesso a mais que alguns conjuntos de *workstations*, o que, muitas vezes é menos que o desejado (ANDRADE, 2002).

A execução simultânea de aplicações é coordenada por um nó central que escalona as tarefas nas máquinas que executam cada aplicação. Esse nó central é chamado *home machine* e as demais máquinas são chamadas *grid machines.* A *home machine* possui um escalonador, o *scheduler*, que recebe do usuário a descrição das tarefas a executar, define onde cada tarefa será executada, submete e monitora cada uma delas de acordo com sua estratégia de escalonamento. Cada máquina do *grid* executa uma tarefa de cada vez. Normalmente, as máquinas do *grid* não têm um sistema de arquivos comum e não são idênticas com relação tanto ao *hardware* como ao *software* (CARDOSO; MATTOSO e MATTOS, 2007).

O MyGrid tem como objetivo ser simples, completo e seguro. A utilização do MyGrid tenta ser o mais próxima possível de uma solução pronta (*plug-and-play*). É um sistema seguro por apresentar proteção a vulnerabilidades no ambiente computacional do usuário. Um conjunto mínimo de serviços, definidos pelo MyGrid, deve ser oferecido por cada máquina pertencente ao g*rid.* Os serviços são: (i) disparo (*start-up*) de tarefas nas máquinas do *grid*; (ii) cancelamento de tarefas que estejam em execução; (iii) transferência de arquivos das máquinas do *grid* para o nó central (*home machine*) e vice-versa (CARDOSO; MATTOSO e MATTOS, 2007).

O MyGrid também pode ser utilizado para prover serviços que auxiliem a gerência da proveniência de dados, utilizando recursos de *Web* semântica, associados a ambientes de *grid,* porém, não permite determinar diferenças entre experimentos e melhorar experimentos futuros. Sua interoperabilidade com outros sistemas é limitada devido ao uso de padrões não abertos para a definição de *workflows* (BALBI; PIRES e MATTOSO, 2005).

O MyGrid apresenta dois componentes responsáveis pela gerência de dados, o MIR (MyGrid *Information Repository*) e o KAVE (*Knowledge Annotations and Verification of Experiments*). O MIR é o responsável pelo armazenamento dos dados dos usuários, como, por exemplo, resultados obtidos em experimentos *in silico* e metadados agregados a cada experimento. É habilitado como um serviço podendo ser acessado por outros componentes do MyGrid para inserir, excluir ou atualizar dados do repositório. O MIR foi baseado no *myGrid Information Model* (IM), que é composto por quatro "pilares"

básicos: (i) projeto de experimentos científicos; (ii) pessoas e organizações que executam esses experimentos; (iii) dados biológicos e de proveniência obtidos nos experimentos; e (iv) operações computacionais envolvidas nos experimentos.

4.1.1 FreeFluo

O FreeFluo é a máquina de *workflow* do MyGrid e funciona como uma ferramenta de gerenciamento de *workflows* para serviços *web.* Suporta um objeto modelo de *workflow* na forma de um grafo direto, no qual cada nó possui uma máquina de estados que define seu ciclo de vida. A sincronização e a transição de estados são controladas com a transmissão de mensagens entre os nós de acordo com o progresso da execução (CARDOSO; MATTOSO e MATTOS, 2007).

Esta ferramenta possui um *parser* de linguagem de *workflow* capaz de converter uma especificação textual de *workflow* para uma representação interna do objeto. Possui também um gerenciador que invoca serviços no ambiente de tempo real e trata os dados passados entre os serviços (CARDOSO; MATTOSO e MATTOS, 2007).

4.1.2 SCUFL

SCUFL (*Simple Conceptual Unified Flow Language*) é um linguagem própria de definição de *workflows*, definida pela equipe do MyGrid, que provê um nível bem mais alto de visão dos *workflows* do que WSFL. Além de ser mais simples de escrever *workflows* a mão utilizando SCUFL, essa linguagem tem a capacidade de representar os *workflows* escritos em WSFL e mais outros que não podiam ser representados utilizando esta última (CARDOSO; MATTOSO e MATTOS, 2007).

Possui recursos para facilitar o processo de proveniência de dados e tolerância à falhas. Como o MyGrid trabalha com serviços *Web*, que podem estar indisponíveis em um dado momento, a tolerância a falhas é muito importante. Alguns metadados que representam dados como quantidade de tentativas de conexão e tempo entre duas tentativas consecutivas, são incorporados à linguagem SCUFL para implementar o mecanismo de tolerância a falhas (CARDOSO; MATTOSO e MATTOS, 2007).

Um documento SCUFL possui "<*processor*>", "<*link*>", "<*source*>", "<*sink*>" e "<*coordination*>" como suas tags (CARDOSO; MATTOSO e MATTOS, 2007).

4.2 Taverna

É um ambiente integrado para desenvolvimento e execução de *workflows*. Surgiu como um componente do projeto piloto do MyGrid, com o propósito de facilitar as atividades de bioinformática capturando métodos científicos como modelos de processos formais, múltiplos e escaláveis, com suporte a grandes conjuntos de dados (*data sets*) (CARDOSO; MATTOSO e MATTOS, 2007; ALBRECHT, 2007).

Entre as vantagens do Taverna pode-se destacar a boa documentação e lista de discussão ativa. É um ambiente multiplataforma, executando em *Linux*, *Mac* e *Windows*, e apresenta código aberto (ALBRECHT, 2007). Além disso, o Taverna permite ao usuário construir *workflows* complexos a partir de componentes localizados na máquina local ou em máquinas remotas, além de permitir a execução desses *workflows* sobre os dados locais, com possibilidade de visualização dos resultados. Esta ferramenta oferece uma linguagem própria para definição e edição de *workflows,* a SCUFL *(Simple Conceptual Unified Flow Language)*, derivada do MyGrid, e uma interface gráfica que facilitam a construção, execução e edição de *workflows* sobre recursos computacionais distribuídos (TAVERNA, 2007). O uso de uma linguagem própria para a definição de *workflows*, pode ser uma boa opção por se ter uma linguagem projetada especificamente para a aplicação científica, apresentando todas as principais estruturas necessárias à construção de *workflows* científicos. Por outro lado, essa linguagem específica pode dificultar o compartilhamento e reaproveitamento de *workflows*, por não ser um padrão já disseminado na comunidade de computação, como a BPEL, por exemplo. Por exemplo, para que um outro cientista utilize os *workflows* definidos com SCUFL por uma outra pessoa, ele deve conhecer essa linguagem ou possuir o Taverna instalado em sua máquina. Esta situação torna o compartilhamento uma atividade não tão direta.

Neste ambiente não há padrões rígidos no desenvolvimento de *workflows*, o que facilita o desenvolvimento de *workflows ad-hoc*. Esta ausência de padrões pode ser pouco interessante para usuários pouco experientes na definição de *workflows*, pois o sistema o deixa livre para criar o que quiser, sem direcioná-lo na criação. Se o usuário não tiver experiência na definição de *workflows*, ele pode não conseguir criar a estrutura desejada. O foco para a definição de *workflows* está no fluxo dos dados, e há possibilidade de especificação de mecanismos de tolerância a falhas, com um nível de abstração apropriado (CARDOSO; MATTOSO e MATTOS, 2007). O foco da definição mantido no fluxo de dados pode ser interessante quando não se deseja variar os dados de forma independente da estrutura de execução. Porém, se for necessário variar os dados para uma mesma estrutura de execução fixa, os resultados obtidos podem estar comprometidos, pois os dados

influenciarão na execução do *workflow*, e sua variação provoca variações na estrutura do mesmo.

O Taverna oferece aos usuários recursos gráficos para construção, edição e exibição dos *workflows*. A interface gráfica, basicamente, consiste de três painéis (CARDOSO; MATTOSO e MATTOS, 2007; TAVERNA, 2007):

- *Advanced Model Explorer*: É uma visão geral do *workflow* para o usuário em forma de árvore. Demonstra processadores e serviços presentes no *workflow*. Os serviços podem ser facilmente importados para *Advanced Model Explorer* a partir do *Available Services Panel,* sendo facilmente incluídos no modelo do *workflow*.
- *Workflow Diagram*: Ferramenta para visualização do *workflow* e seus passos, de forma geral. Auxilia na validação do *workflow* criado.
- *Available Services Panel*: Apresenta uma lista de serviços que podem ser adicionadas ao *workflow*.

A figura 5, apresenta uma visão geral dos três painéis principais que constituem a interface gráfica do Taverna.

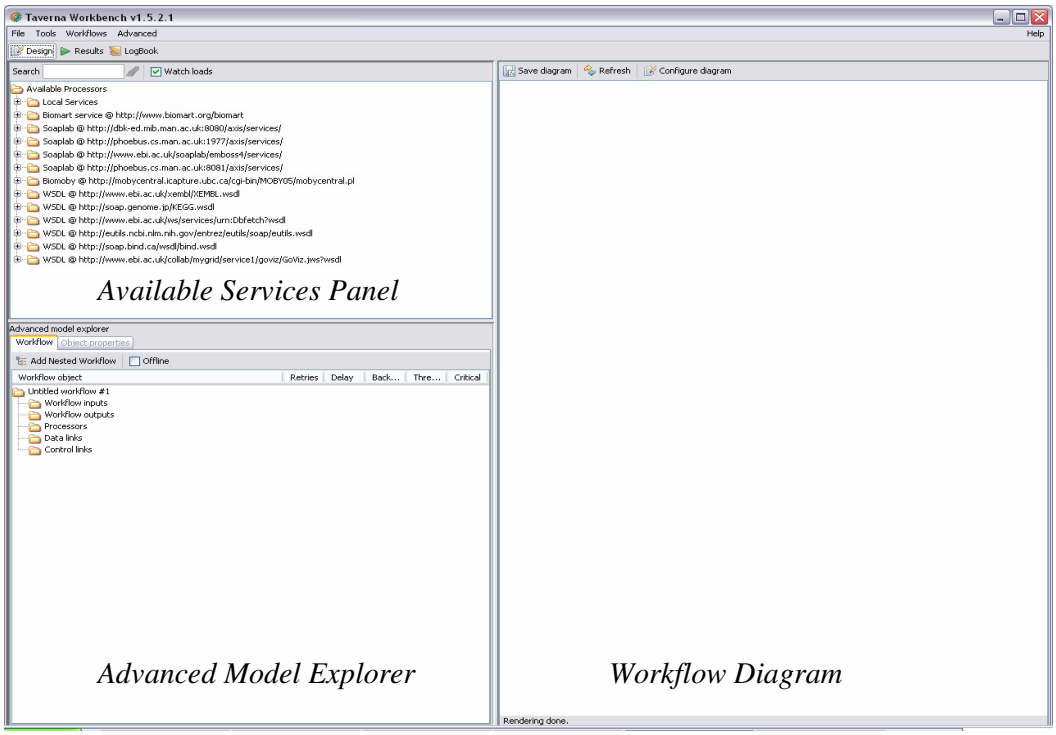

Figura 5 – Interface gráfica do Taverna (FISHER, 2007)

Os *workflows* são executados pela máquina de execução FreeFluo, citada anteriormente. Durante a execução do *workflow*, seu progresso pode ser monitorado por um painel específico do Taverna, o *Enactor invocation panel*. Os serviços são invocados de acordo com o modelo do *workflow*, e seus *status* vão se modificando de *scheduled* (agendado, não executado) para *invoking* (em execução) e, por fim, para *complete* (executado). Se um serviço falha, isso também é possível de ser acompanhado pelo painel de monitoração do Taverna. Os resultados finais do *workflow* também podem ser visualizados no painel de monitoramento de execução. Cada saída diferente do *workflow* é gravada separadamente e pode ser salva como um arquivo XML ou um arquivo do *Excel.* Os resultados podem ser salvos separadamente ou juntos, em um mesmo formato. No Taverna, um *workflow* é executado por uma instância do *workflow*, que é representada por um arquivo XML no formato Xscufl (TAVERNA, 2007).

No Taverna, o transporte e monitoramento de dados são manipulados automaticamente, e todo o conjunto de componentes de um *workflow* é executado sem nenhuma intervenção externa. Isso permite ao usuário iniciar a execução de um *workflow* e, sobretudo, no caso de *workflows* com longos períodos de execução, ir executar alguma outra tarefa, ao mesmo tempo. Sem um sistema de *workflow*, a invocação de componentes externos à máquina local se dá de forma manual, por exemplo, e isso toma o tempo do usuário, pois ele precisa monitorar toda a execução do *workflow* para fazer essas invocações quando necessário (TAVERNA, 2007). Porém, essa característica pode não ser muito interessante para o usuário que deseja executar o *workflow* manualmente, modificando componentes para analisar o comportamento dos dados, por exemplo. O usuário, muitas vezes, também necessita de fazer interrupções no execução do *workflow* para ter acesso a resultados intermediários, porém, se tudo ocorre de forma automática, sem seu controle, essa atividade não pode ser executada.

Além das facilidades com relação à invocação automática de componentes, o Taverna também admite que esses componentes sejam locais ou remotos, o que permite que qualquer usuário, com um PC razoavelmente moderno, acesse qualquer supercomputador em qualquer parte do mundo, com a utilização de poucos ou nenhum recurso administrativo. Embora as máquinas modernas apresentem recursos computacionais significantes, ainda existem muitos algoritmos que necessitam de recursos computacionais em escala industrial. Usando componentes remotos, o usuário pode tirar vantagem dos recursos computacionais oferecidos pelo provedor do serviço que está sendo utilizado remotamente. O resultado final desta condição é que o *workflow* pode ser

completado bem mais rápido do que se fosse executado completamente na máquina local do usuário (TAVERNA, 2007).

No Taverna, é possível configurar tolerância a falhas na definição de um *workflow*. É possível repetir a execução após falhas ou desviar a execução para um ou vários outros caminhos alternativos quando todas as repetições possíveis já tiverem sido esgotadas. É possível definir quantas vezes um processo será repetido, após uma falha, até que sua execução ocorra da forma esperada ou até que seu curso seja desviado para caminhos alternativos, que também poderão ser definidos. O tempo de atraso entre as repetições também é configurável, o que pode ser muito útil na utilização de recursos remotos, que podem estar indisponíveis em um dado instante e podem estar disponíveis num instante seguinte (TAVERNA, 2007). O tempo de atraso deve ser muito bem calculado, antes de ser configurado, pois, se for grande demais, pode alongar demasiadamente o tempo de execução de *workflow* e, se for muito curto, pode não ser suficiente para conseguir invocar um serviço remoto indisponível temporariamente.

Esta ferramenta apresenta a propriedade de *Threads*, que determina quantas instâncias concorrentes poderão ser usadas durante uma interação. Essa propriedade é interessante quando se tem a necessidade de executar várias instâncias de um processo e se dispõe de diversos processadores. Se uma instância apresenta um tempo de execução X, ao executar N instâncias deste tipo sequencialmente, o tempo gasto será N.X, porém, se N processadores forem utilizados para executar essas instâncias simultaneamente, o tempo gasto será X, aumentando assim, o desempenho na execução do processo (TAVERNA, 2007). A definição da quantidade de *threads* também deve ser muito bem estudada, pois, se muitas instâncias forem executadas simultaneamente em uma máquina com pouca capacidade de processamento, o efeito de otimização que se espera das *threads* pode não ocorrer e ainda o efeito contrário, de atraso no prcessamento, pode ocorrer. Por outro lado, se for definida uma quantidade muito pequena de instâncias a serem executadas simultaneamente, em um ambiente de grande capacidade de processamento paralelo, todo o potencial do ambiente não será corretamente aproveitado.

O Taverna possibilita uma visão gráfica do diagrama do *workflow*. Dados de entrada, de saída e processos aparecem como caixas coloridas, com setas entre elas, que representam *links* de controle de dados e de execução. É possível configurar a forma de apresentação e o nível de detalhes do diagrama. O diagrama final pode ser salvo em diferentes formatos. Este recurso facilita a visualização rápida da estrutura do *workflow*. As cores dos componentes do *workflow* apresentam significado (TAVERNA, 2007).

É possível definir *workflows* aninhados (*nested workflows*), ou seja, *workflows* que são definidos como partes integrantes de *workflows* maiores e mais complexos. Na visualização é possível ver a estrutura destes *subworkflows* ou vê-los de forma resumida, simplificando assim, a representação do *workflow* maior (TAVERNA, 2007).

O Taverna apresenta um painel de organização dos serviços, que os mostra de forma hierárquica (TAVERNA, 2007).

4.3 VisTrails

Os *workflows* vêem emergindo como um paradigma para a representação de computações complexas. Além do paradigma dos *workflows,* com a explosão do volume de dados científicos, podemos observar uma tendência dos cientistas em utilizar paradigmas de visualização. A crescente demanda por visualização está direcionando o desenvolvimento de novos modelos de sistemas que contribuem para o aumento do poder computacional e para a ampla utilização de *Graphics Processing Units* (GPUs) (BAVOIL *et al.*, 2005).

Neste cenário os sistemas para simples modelagem e execução de *workflows* já não são mais suficientes. São necessários sistemas que ofereçam a infra-estrutura necessária para tarefas exploratórias, de análise dos resultados obtidos. Os *workflows* são, tradicionalmente, utilizados para automatizar tarefas repetitivas, porém, em tarefas exploratórias, as mudanças são imprescindíveis. A análise e exploração de dados são processos interativos (SILVA e FREIRE, 2007).

Alguns fatores dificultam a exploração de dados, podem ser considerados: (i) a proveniência de dados deve ser automatizada e detalhada; (ii) é difícil relacionar a exploração dos dados com seu *workflow* correspondente; (iii) é difícil determinar modificações necessárias para os dados e para a estrutura do *workflow*; e (iv) é difícil compartilhar os conhecimentos adquiridos a partir da exploração de dados. A geração e manutenção do *workflow* são os maiores problemas em processos científicos (SILVA e FREIRE, 2007).

Buscando resolver a maioria desses problemas, o VisTrails, que é um sistema de gerenciamente de *workflows* científicos, foi desenvolvido pela Universidade de Utah, e oferece suporte para a exploração e visualização de dados. Considerando que os *workflow* foram tradicionalmente usados para automatizar tarefas repetitivas, para aplicações que são exploratórias por natureza, muito pouco é repetido, e as mudanças são constantes. Como os

cientistas geram e avaliam hipóteses a partir dos dados que estão sendo estudados, diversos *workflows* diferentes são criados enquanto um *workflow* é ajustado em um processo interativo. VisTrails foi desenvolvido para gerenciar esses *workflows* intermediários. Possibilita a criação, execução e compartilhamento de visualizações complexas, mineração de dados e outras aplicações de análises de dados em larga escala. Por meio do gerenciamento automático de dados, metadados e dos processos de exploração de dados, o VisTrails permite ao usuário se concentrar nas tarefas relacionadas a sua pesquisa e não perder tempo com a organização do grande volume de dados que são manipulados. O VisTrails oferece uma infra-estrutura que pode ser combinada ou mesclada com outros sistemas existentes de visualização e de *workflows* (VISTRAILS, 2007).

O VisTrails é um sistema que permite visualizações múltiplas e interativas, simplificando a criação e a manutenção de *pipelines* de visualização, e otimizando sua execução. Esses fatores geram uma infra-estrutura que pode ser combinada com outros sistemas de visualização e bibliotecas existentes. Este sistema é útil para diversas aplicações de visualização. Ele pode ser usado para explorar os parâmetros em uma visualização, repetir visualizações com dados diferentes, e para comparar diferentes técnicas de visualização. O principal componente do VisTrails é o *visualization trail* (*vistrail*), uma especificação formal de um *pipeline* (BAVOIL *et al.*, 2005).

O *visualization trail* (ou *vistrail*) é uma especificação formal de um *pipeline*. A especificação de um *vistrail* consiste em uma seqüência de operações usadas para gerar uma visualização. É gravada a proveniência para as imagens geradas, que pode ser usada para repetir, de maneira automática, a geração da imagem. Um *vistrail* é armazenado como um documento XML. Um elemento *vistrail* contém um conjunto de módulos, conexões entre esses módulos e um elemento opcional de anotação. Cada módulo contém uma ou mais funções, sendo que cada função tem um retorno e um conjunto de parâmetros e as conexões capturam as dependências dos dados entre módulos diferentes (BAVOIL *et al.*, 2005).

Abrangendo a criação, execução e compartilhamento de um grande número de *workflows* complexos, o VisTrails gerencia os dados, os metadados e a exploração do processo. Porém, este sistema não permite a configuração de tolerância a falhas na definição dos *workflows*, assim como também não oferece recurso de validação de *workflows* nem permite a intervenção do usuário na execução, com a conseqüente obtenção de resultados intermediários de acordo com a escolha do usuário. O VisTrails apresenta três camadas de metadados: (i) os dados de evolução do *workflow*; (ii) os dados de

definição do *workflow*; e (iii) os dados resultantes da execução do *workflow*. Para explorar os dados de proveniência, existe uma linguagem para consultas. Este sistema não tem o intuito de substituir os sistemas de visualização de *workflows* científicos, ele oferece infra-estrutura para auxiliar e reforçar estes sistemas (SILVA e FREIRE, 2007).

Diferente dos sistemas baseados em fluxos de dados, no VisTrails há uma clara separação entre a especificação de um *pipeline* e suas instâncias de execução. A especificação serve como um modelo e o usuário pode executá-la utilizando diferentes parâmetros (BAVOIL *et al.*, 2005).

O VisTrails vai além da especificação do *vistrail,* identificando e evitando operações redundantes. É gerada uma otimização que é especialmente útil para visualizações múltiplas, em que o *vistrail* pode conter seqüências semelhantes. Isso pode ocorrer, por exemplo, quando variações de um mesmo *trail* são executadas. Por meio do reuso de resultados já computados para seqüências parecidas de diferentes *pipelines*, o VisTrails gera uma melhora de interatividade (BAVOIL *et al.*, 2005).

No VisTrails, são usadas *XML query languages* como, por exemplo, XPath e XQuery para consultar a especificação e as instâncias do *vistrail*. Usando uma *XML query language*, o usuário pode, por exemplo, consultar um conjunto de *vistrails* armazenados para localizar um que seja adequado para uma determinada tarefa. Além dessas facilidades de uso, o *framework* VisTrails é extensível e pode ser usado por *workflows* científicos em geral (BAVOIL *et al.*, 2005).

Os principais objetivos do sistema são: (i) a criação de uma infra-estrutura que mantenha a proveniência de um grande número resultados (imagens) de visualizações de dados; (ii) o oferecimento de um *framework* geral para a execução eficiente de um grande número de *pipelines* de visualização potencialmente complexos; e (iii) o oferecimento de uma interface com o usuário que simplifique a comparação entre diversas visualizações. O VisTrails busca construir uma infra-estrutura extensível e flexível que seja capaz de reforçar e ser combinada com os sistemas já existentes (BAVOIL *et al.*, 2005).

Como no VisTrails, existe uma clara separação entre a especificação do *pipeline* e suas instâncias de execução, a correspondência entre os valores dos parâmetros e a visualização resultante é mantida no sistema; e pela combinação deste fator com o mecanismo que controla as versões das especificações de *pipeline*, um histórico completo do *pipeline* pode ser mantido, por meio de um registro de proveniência das visualizações. Pela visualização da especificação de um *pipeline* como um modelo e permitindo ao usuário especificar conjuntos de entradas para um determinado *pipeline*, é oferecido um

mecanismo para a geração de um grande número de visualizações. Como já foi dito, o VisTrails otimiza a especificação do *vistrail* identificando e evitando operações redundantes. Essa otimização gera uma melhora substancial de desempenho, permitindo a exploração de um grande número de visualizações simultaneamente. Quando variações do mesmo *pipeline* precisam ser executadas, *speedups* substanciais podem ser obtidos armazenando (*caching*) os resultados de execuções subseqüentes dos *pipelines* (BAVOIL *et al.*, 2005).

No VisTrails, históricos de sessão podem ser construídos a partir de informações armazenadas em um conjunto de instâncias *vistrail*, facilitando a repetição de operações. Uma instância *vistrail* guarda informações de proveniência de suas imagens derivadas. Essas informações incluem nome e localização dos arquivos com dados de entrada. Para garantir a possibilidade de reprodução das execuções, o sistema faz cópias dos dados de entrada. Um banco de dados é utilizado para armazenar os metadados, e os dados de entrada são armazenados em sistemas de arquivos. Um problema com essa forma de armazenamento dos dados de entrada é que os arquivos de armazenamento podem ser movidos, apagados ou modificados. Além disso, é oferecida uma infra-estrutura que permite o uso efetivo de diversas técnicas de visualização, o que permite ao usuário explorar e entender as informações subjacentes (BAVOIL *et al.*, 2005).

O *framework* VisTrail possui um módulo gerenciador de *cache* (*Vistrail Cache Manager* – VCM) que agenda a execução de módulos nos *vistrails.* Quando os passos de um *vistrail* são executados, o VCM armazena seus resultados, uma assinatura do módulo que inclui os nomes e valores dos parâmetros, juntamente com um *handle* para os dados de saída do módulo. Quando o VCM identifica sub-redes em um *vistrail* que já foram processadas, localmente, a computação possivelmente dispendiosa é transformada em *cache*. Para maximizar a utilidade do *cache*, os resultados armazenados são compartilhados entre os *vistrails.* Isso permite a eliminação de processamentos redundantes em seqüências correspondentes de *vistrails* diferentes (BAVOIL *et al.*, 2005).

Além do módulo de gerência de *cache*, existem também o *Vistrail Player* (VP), que é um interpretador bem simples, que recebe como entrada um arquivo XML para uma instância do *vistrail* e a executa; o *Vistrail Builder* (VB), que apresenta uma interface gráfica para a criação e edição de *vistrails*, escreve e lê *vistrails* no mesmo formato XML utilizado pelos outros módulos do sistema, além de utilizar o mesmo paradigma de módulos e conexões que os sistemas de fluxo de dados; e a planilha de visualização (V*isualization Spreadsheet* – VS) do VisTrails, que permite ao usuário comparar os

resultados de múltiplos *vistrails,* por meio de um conjunto de janelas de visualização separadas (BAVOIL *et al.*, 2005). Essa comparação de resultados é feita pelo usuário, pois o sistema não apresenta um mecanismo de comparação, com a apresentação de relatórios de diferenças ou semelhanças entre *vistrails* diferentes.

O *layout* da planilha de visualização do VisTrails oferece uma utilização eficiente do espaço da tela. As células de visualização podem executar diferentes *vistrails* e também podem usar diferentes parâmetros para a mesma especificação de *vistrail*. Para garantir uma execução eficiente, todas as janelas compartilham a mesma *cache*. O *Vistrail Player* é multiplataforma, rodando em *Windows*, *MAC* e *Linux* (BAVOIL *et al.*, 2005). Na figura 5 podemos observar as janelas do VisTrails, com seus módulos funcionais.

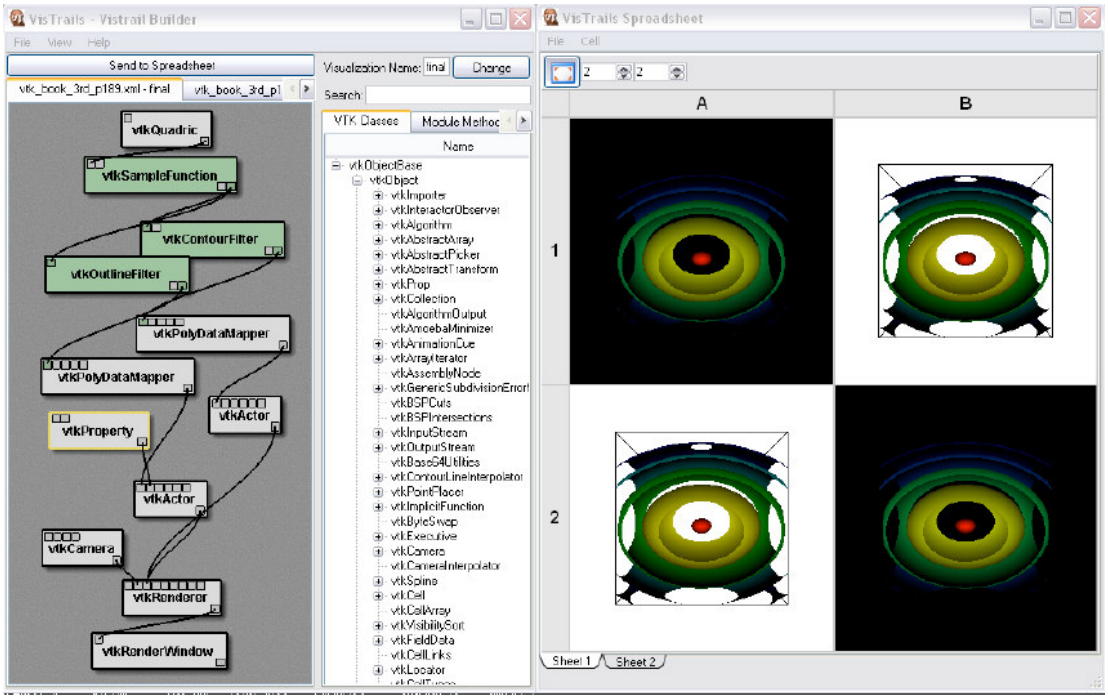

Figura 6 – Janelas do VisTrails (VISTRAILS, 2007)

Usando um *vistrail*, os usuários podem resgatar explorações anteriores; aplicar facilmente operações idênticas a um novo conjunto de dados, sem ter que refazer longas seqüências de operações; compartilhar seus resultados com outros usuários; testar novos códigos a partir da comparação entre visualizações geradas a partir de diferentes versões de uma simulação. Por meio do *caching* de resultados intermediários e do compartilhamento de computações complexas, é possível também reduzir consideravelmente seu tempo de

resposta, possibilitando ao usuário explorar, de forma eficiente, grandes volumes de dados (BAVOIL *et al.*, 2005).

4.4 Kepler

Kepler é um *software* para análise e modelagem de dados científicos, que tem como objetivo principal reduzir o esforço necessário para a criação de modelos executáveis, por meio do uso de uma representação visual de processos. Essas representações, que também podem ser chamadas de *workflows* científicos mostram o fluxo de dados através de análise discreta e modelagem de componentes (KEPLER, 2007; LUDÄSCHER *et al.*, 2004).

É um *software open-source* e multiplataforma, podendo rodar em plataformas *Windows*, *Linux* e *Machintosh* (LUDÄSCHER *et al.*, 2004). Embora o Kepler esteja sendo desenvolvido para se tornar uma aplicação estável e totalmente desenvolvida, este *software* é apenas um projeto de pesquisa, que vem sendo desenvolvido em um trabalho contínuo. Os usuários do Kepler podem contribuir com sugestões de melhora, assim como também podem auxiliar os desenvolvedores reportando *bugs* e outros problemas que possam vir a ocorrer. A pesquisa e o desenvolvimento no Kepler se beneficiam de interações e colaborações com outros grupos (LUDÄSCHER *et al.*, 2004). A participação dos usuários no desenvolvimento da aplicação ajuda a adaptar o projeto às necessidades reais dos cientistas. Para instalar e utilizar o Kepler é necessário ter o Java 1.5 ou superior instalado (KEPLER, 2007).

Workflows científicos são ferramentas flexíveis para acesso a dados científicos e execução de análises complexas nesses dados. Cada *workflow* consiste de passos que podem envolver acesso a bancos de dados, análise e mineração de dados, e computação intensiva em *clusters* de computadores de alto desempenho. Cada passo do *workflow* é representado por um "ator" (*actor*), que, na interface gráfica do Kepler, é um componente que pode ser arrastado e incluído em um *workflow*. Atores conectados formam um *workflow*, permitindo aos cientistas analisar e mostrar dados durante seu processamento, modificando parâmetros, se necessário, e re-executando e reproduzindo resultados experimentais (KEPLER, 2007).

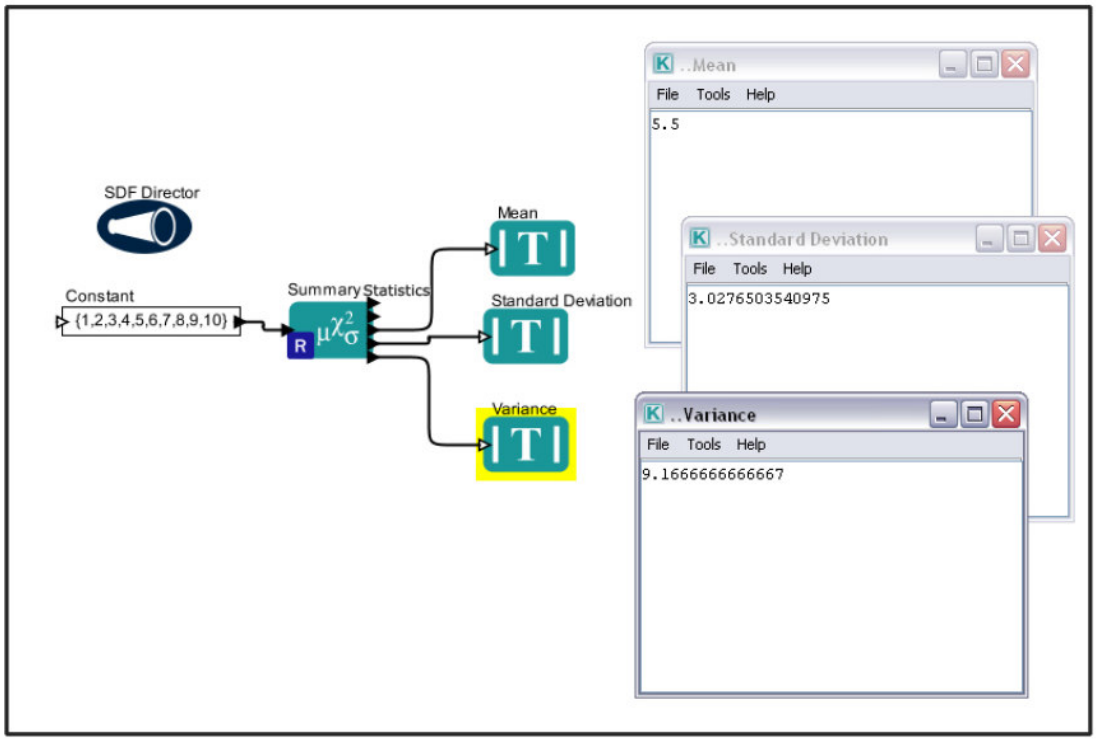

Figura 7 - Exemplo de um *workflow* científico desenvolvido no Kepler (KEPLER, 2007)

Workflows podem representar modelos teóricos ou análises, podem ser simples e lineares, ou podem ser complexos e não lineares. Um dos benefícios do *workflow* científico é que ele pode ser aninhado, o que significa que um *workflow* pode conter *subworkflows* que executam tarefas embutidas. Um *workflow* aninhado (também conhecido como um ator composto) é um componente reusável que executa uma tarefa potencialmente complexa. *Workflows* científicos no Kepler têm acesso aos mais atuais benefícios da computação em grade (permitindo acesso a recursos computacionais distribuídos como dados e serviços de computação) sem revelar a complexidade escondida sob este tipo de tecnologia. As tarefas de processamento de dados em baixo nível são automatizadas, permitindo que os cientistas mantenham seu foco nas questões científicas de seu interesse (KEPLER, 2007).

Um *workflow* também fornece documentação de todos os aspectos de uma análise, a representação visual dos passos da análise, a possibilidade de trabalhar através de múltiplos sistemas, além de permitir a reprodução de um projeto de maneira simples, sem muitos esforços, e também permitir o reuso de parte ou de todo o *workflow* em um projeto diferente. No Kepler é possível que o usuário modele, analise, e mostre os dados por meio de uma interface simples. O Kepler permite aos cientistas criar seus próprios *workflows*

científicos executáveis por meio da simples inclusão ou exclusão de componentes em uma área de criação de *workflows,* e conexão dos componentes para construir um fluxo de dados específico, criando um modelo visual da porção analítica de uma pesquisa. É possível representar a visualização completa do *workflow* de uma forma fácil de entender como os dados fluem de um componente para outro. Dessa forma, a definição do *workflow* está focada no fluxo dos dados, o que torna a estrutura de execução do *workflow* dependente dos destes, dificultando a variação de dados para uma mesma estrutura. O *workflow* resultante pode ser salvo em formato texto e, dessa forma, compartilhado com outras pessoas (KEPLER, 2007).

Uma gama de componentes prontos está incorporada como padrão no Kepler, incluindo componentes matemáticos, estatísticos, componentes para processamento de sinais e componentes para inserção, manipulação e exibição de dados. O usuário também pode criar novos componentes ou utilizar componentes já existentes em outros. Usuários iniciantes em computação podem criar *workflows* com componentes padrão, ou modificar os *workflows* existentes para que eles atendam suas necessidades. Componentes pré-programados do Kepler podem ser facilmente incluídos em uma representação visual já existente de um *workflow*. Os usuários avançados também podem utilizar o Kepler para representar programas e análises complexas, podendo compartilhá-los de forma simples e compreensível (KEPLER, 2007).

Para cada *workflow* diferente que for aberto ou criado, uma nova janela do Kepler será aberta. Múltiplas janelas permitem ao usuário trabalhar em diversos *workflows* simultaneamente e comparar, copiar e colar componentes entre os *workflows* abertos (KEPLER, 2007).

O Kepler provê acesso a repositórios de dados, recursos computacionais e bibliotecas de *workflows* distribuídos geograficamente (KEPLER, 2007). A partir da URL da descrição de um serviço *web*, um ator genérico de serviço *web* do Kepler é capaz de instanciar operações particulares especificadas na descrição do serviço. Depois da instanciação, este ator pode ser incorporado a um *workflow* científico como se ele fosse um componente local. Os dados de entrada e saída do serviço *web*, especificados na descrição WSDL do serviço, são representados pelas portas de dados de entrada e de saída do ator (LUDÄSCHER *et al.*, 2004).

Para a construção de *workflows* científicos no Kepler, são utilizados componentes básicos customizáveis. São eles: atores (*actors*), diretores (*directors*) e parâmetros (*parameters*), além das relações (*relations*) e portas (*portas*), que permitem a comunicação

entre os componentes (KEPLER, 2007). O sistema não oferece mecanismo de validação do *workflow* nem agendamento de execução para o mesmo.

Um diretor controla (ou dirige) a execução de um *workflow*. Os atores recebem suas instruções de execução do diretor. Em outras palavras, o ator especifica o que será executado e o diretor especifica quando será executado. Todo *workflow* deve ter um diretor que controla sua execução usando um modelo particular de computação. Cada modelo de computação no Kepler é representado pelo seu próprio diretor. Um pequeno conjunto de diretores mais comumente utilizados vem acoplado ao Kepler, mas outros podem ser acessados remotamente quando necessário. Em *workflows* mais complexos é possível se ter diferentes diretores em diferentes níveis. Kepler fornece um grande conjunto de atores predefinidos para criação e edição de *workflows* científicos, porém, novos atores podem ser adicionados ao Kepler para uso exclusivo de um usuário podendo ser utilizados por outros (KEPLER, 2007).

Atores compostos (*composite actors*) são coleções ou conjuntos de atores reunidos para executar operações mais complexas. Eles podem ser usados em *workflows*, atuando essencialmente como *workflows* aninhados ou *sub-workflows*. Um *workflow* completo pode ser representado como um ator composto e incluído como um componente em um outro *workflow* (KEPLER, 2007).

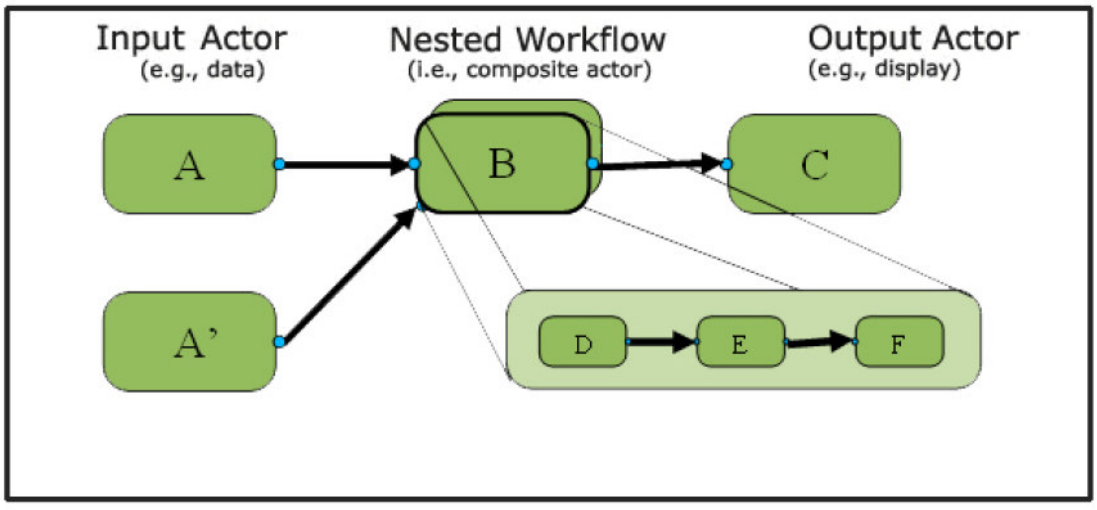

Figura 8 - Representação de um *workflow* aninhado (KEPLER, 2007)

Cada ator em um *workflow* pode conter uma ou mais portas, que são usadas para consumir ou produzir dados, e para comunicação com outros atores do *workflow*. Atores são conectados em um *workflow* por meio de suas portas. A ligação que representa o fluxo

de dados entre a porta de um ator e outro ator é chamada canal. Existem três tipos de portas: (i) *input port*: porta para os dados consumidos pelo ator; (ii) *output port*: para os dados produzidos pelo ator; e (iii) *input/output port*: para dados consumidos e produzidos pelo ator (KEPLER, 2007).

As relações permitem aos usuários ramificar um fluxo de dados, e dessa forma os dados podem ser enviados para múltiplos lugares no *workflow*. Por exemplo, se for necessário os dados de saída de um ator operacional para um outro ator operacional, e para um ator de exibição para exibir aqueles dados, pelo uso de relação no canal de saída, o usuário pode direcionar os dados para ambos os lugares simultaneamente (KEPLER, 2007). Este recurso permite, de forma implícita, a configuração de um mecanismo de tolerância a falhas, porém, de forma precária. Ao definir o *workflow*, o usuário tem que definir o mecanismo de tolerância manualmente, sem o auxílio do sistema, que não oferece esse recurso de forma explícita.

Parâmetros são valores configuráveis que podem ser adicionados ao *workflow,* ou a diretores, ou atores (KEPLER, 2007). *Workflows* científicos são criados e editados no Kepler por meio de uma interface simples de "arrastar e soltar". É possível observar esta interface na figura 9. Apenas por meio da interface é possível fazer uma validação do *workflow*. O usuário tem que utilizar a estrutura do *workflow*, mostrada por meio da interface e então, fazer ele mesmo a validação, pois o sistema não oferece este recurso de forma automática.

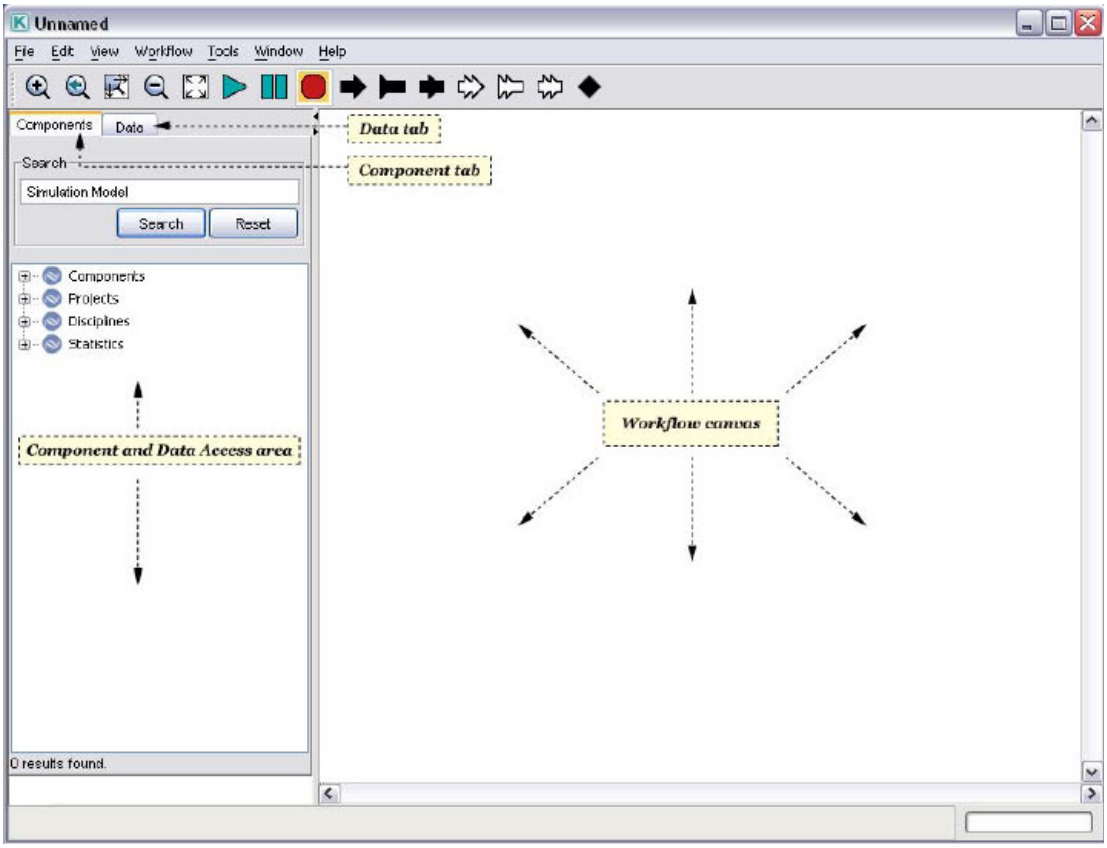

Figura 9 – Uma janela vazia do Kepler, com suas principais seções indicadas (KEPLER, 2007).

As abas de componentes e de dados (*Component tab* e *Data tab*), como os próprios nomes sugerem, mostram componentes e dados. A aba de componentes contém uma biblioteca dos componentes do *workflow*. Os componentes são mostrados em categorias. A aba de dados contém um mecanismo de busca para localizar e usar conjuntos de dados (que podem estar disponíveis na *web*) (KEPLER, 2007).

KEPLER (2007) fornece um mecanismo de busca para localizar dados (em um *Grid*) e componentes (no sistema local).

4.5 Análise comparativa das principais ferramentas: Taverna, VisTrails e Kepler

Como foi apresentado no início deste capítulo, é desejável que um sistema de gerência de *workflows* inclua os processos, dados e recursos normalmente usados e ofereça mecanismos de extensibilidade para acomodar novos processos, dados e recursos; permita ao usuário a definição e redefinição de *workflows*, permitindo a definição de propriedades

dos processos, dados e recursos facilitando a escolha do pesquisador de acordo com sua necessidade; ofereça ferramentas de validação do *workflow* definidas pelo pesquisador; ofereça o recurso de tratamento de exceção; seja capaz de otimizar e executar o *workflow* definido pelo pesquisador; possibilite a interferência do usuário na execução do *workflow*; ofereça agendamento da execução do *workflow*; gere e armazene, automaticamente, metadados sobre o *workflow*.

Além dessas características, também podem ser consideradas como funcionalidades interessantes, que o sistema seja multiplataforma, ofereça suporte ao trabalho com grande volume de dados, permita a visualização de resultados intermediários e ofereça mecanismo de consulta dos metadados armazenados.

É possível verificar algumas dessas características nas ferramentas apresentadas ao longo deste capítulo. Todas elas são multiplataforma, oferecem suporte ao trabalho com grande volume de dados, apresentam extensibilidade, permitindo a inclusão de novos processos, recursos e dados, além de permitirem definição e redefinição de *workflows*, com mecanismo para execução dos mesmos.

O VisTrails, assim como o Kepler, traz com sua configuração um conjunto de componentes padrão para a definição de um *workflow*. Taverna e Kepler permitem a construção de mecanismos para o tratamento de exceções, sendo que o Taverna apresenta esse mecanismo de forma mais automática e explícita, já o Kepler permite que essa construção seja feita, de forma implícita, a partir da combinação de outros recursos.

VisTrails e Taverna geram e armazenam metadados sobre os *workflows* executados. O VisTrails também é capaz de gerenciar estes dados e ainda oferece uma estrutura de consulta para que o usuário tenha um acesso facilitado a dados de proveniência. Além dessa capacidade, o VisTrails também oferece um incrível controle de versão dos *workflows* que vão sendo gerados e modificados. O agendamento e validação dos *workflows* são funcionalidades que não são oferecidas, de forma automatizada, por nenhuma das ferramentas apresentadas.

A partir das informações obtidas no estudo de cada uma dessas ferramentas, é possível notar que algumas funcionalidades podem ser encontradas em uma ferramenta e não em outra, o que mostra que ainda não é possível se ter a uma ferramenta completa, que atenda a todas as necessidades de todos os perfis de usuários. Para ter acesso a todos os recursos de que se necessita, o usuário tem que fazer uso de diversas ferramentas, combinando a facilidades obtidas pelo uso cada uma delas, de acordo com seus interesses.

Capítulo 5

Conclusões

É cada vez mais comum no meio científico que grupos de pesquisa distintos colaborem entre si compartilhando dados, análises, técnicas e recursos computacionais. Os recursos computacionais, ao serem aplicados às atividades de pesquisa científica, estão colaborando com o crescimento da produção científica nas mais variadas áreas de conhecimento. Esses recursos, além de possibilitarem a colaboração entre os cientistas, permitem que eles mantenham seu foco nas atividades de pesquisa sem se preocupar com formas de armazenamento e compartilhamento de recursos. Algumas áreas de pesquisa, que geram grande quantidade de dados, utilizam recursos computacionais de alto desempenho para atender à sua demanda de armazenamento, processamento e análise dos dados obtidos.

Nesse contexto, o termo *e-science* é geralmente empregado para descrever o desenvolvimento de infra-estruturas de serviços de *software* capazes de prover acesso a facilidades remotas, recursos computacionais remotos, armazenamento de informações em bancos de dados dedicados e disseminação e compartilhamento de dados, resultados e conhecimento. Em *e-science,* são utilizadas tecnologias como a proveniência de dados, tecnologias de integração de informações, *grids* computacionais e serviços *Web* para o apoio à realização de experimentos científicos. Essas tecnologias e os principais conceitos ligados a elas, são apresentados no capítulo 2 desta monografia, possibilitando um melhor entendimento desse contexto de pesquisas científicas que são realizadas com o apoio de recursos computacionais.

Um *workflow* para *e-science* é um *workflow* voltado para explicitar representações de processos experimentais científicos, geralmente de natureza colaborativa. O conceito de *workflow*, assim como os detalhes comuns e as disparidades entre os *workflows* científicos e de negócio, são apresentados no capítulo 3, oferecendo uma noção de como essas tecnologias funcionam na prática, sem deixar de mostrar suas principais características conceituais. Além de especificar o que são e como funcionam os *workflows*, no capítulo 3, também é apresentada uma linguagem para a definição de *workflows*, BPEL. São apresentadas as principais construções dessa linguagem, o que permite um melhor entendimento de como os *workflows* são definidos e modelados.

Para a criação de *workflows* científicos são necessárias ferramentas que possibilitem o seu projeto e composição. A partir da apresentação e análise das principais

funcionalidades desejáveis em ferramentas de gerência de *workflows* e da apresentação de algumas dessas ferramentas no capítulo 4, este trabalho fornece um suporte para a avaliação e possível escolha de ferramentas adequadas ao contexto em que serão utilizadas pelos usuários.

No contexto de *e-science*, podem ser considerados como principais benefícios da implantação de *workflows,* maior eficiência e confiabilidade nos processos experimentais, compartilhamento de informações, cooperação entre diversos grupos de pesquisa e possibilidade de repetição e reprodução de procedimentos. O controle sobre os processos que compõem os experimentos se dá a partir das informações de proveniência, que também podem ser obtidas durante a execução do *workflow*. Os *workflows* científicos podem ser considerados como cruciais na interação dos cientistas com a infra-estrutura computacional aplicada à pesquisa. Eles facilitam as atividades de *e-science* permitindo que os cientistas modelem, executem, verifiquem, reconfigurem e re-executem as análises de dados científicos.

A revisão de conceitos apresentada nesta monografia propicia um melhor entendimento do contexto de *e-science.* Os conceitos apresentados podem ser aplicados na realização de pesquisas reais ligadas a *e-science*.

As ferramentas apresentadas podem ser utilizadas, estudadas e estendidas com novas funcionalidades de modo que passem a atender às necessidades de grupos pertencentes a áreas de conhecimento cada vez mais diversas.

Referências Bibliográficas

ANDRADE, Nazareno. **MyGrid**. 2002. Disponível em: <http://www.lsd.ufcg.edu.br/~nazareno/documentos/acessoEmGrids/node7.html>. Acesso em: 05 novembro 2007.

ALBRECHT, Felipe. **Taverna – Workflows para bioinformática**. 2007. Disponível em: <pihisall.wordpress.com/2007/01/19/**taverna**-**workflows**-para-bioinformatica/>. Acesso em: 12 novembro 2007.

ARAUJO, Renata Mendes de; BORGES, Marcos Roberto da Silva. **Sistemas de Workflow**. 2001. Disponível em: <chord.nce.ufrj.br/cursos/teesi/textos/apostilaJai2001div.pdf>. Acesso em: 05 outubro 2007.

BALBI, Ricardo; PIRES, Paulo; MATTOSO, Marta. **BioProvenance:** Um framework para a proveniência de dados aplicado à bioinformática. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 20, 2005, Uberlândia. Anais... Uberlândia: [s.n.], [2005].

BARTHELMESS, Paulo. **Sistemas de Workflow:** Análise da Área e Proposta de Modelo. Campinas, 1996. 151f. Tese (Mestrado em Ciência da Computação) – Instituto de Computação, Universidade Estadual de Campinas.

BAX, Marcello Peixoto; LEAL, George Jamil. **Serviços *Web* e a evolução dos serviços em TI**. 2001. Disponível em: <http://www.paradigma.com.br/biblioteca/servicos-web/view>. Acesso em: 19 setembro 2007.

BAVOIL, Louis; CALLAHAN, Steven P.; CROSSNO, Patricia J.; FREIRE, Juliana; SCHEIDEGGER, Carlos E.; SILVA, Cláudio T.; VO, HuyT. **VisTrails:** Enabling Interactive Multiple-View Visualizations. 2005. Disponível em: <www.sci.utah.edu/~stevec/papers/vistrails.pdf>. Acesso em: 07 novembro 2007.

CARDOSO, Daniel; MATTOSO, Marta; MATTOS, Amanda. **myGrid e Taverna**: Experimentando Grids e Workflows Científicos. Apresentação da disciplina de Tópicos em Banco de Dados II, COPPE/UFRJ. [2007].

DIGIAMPIETRI, Luciano A.; MEDEIROS, Cláudia B.; SETUBAL, João C. **A Famework Based on Web Services Orchestration for Bioinformatics Workflow Management**. Campinas, 2005. Disponível em: <http://www.funpecrp.com.br/GMR/year2005/vol3-4/wob04_full_text.htm>. Acesso em: 05 setembro 2007.

DIGIAMPIETRI, Luciano Antônio. **Gerenciamento de *workflows* científicos em bioinformática**. Campinas, 2007. 112f. Tese (Doutorado em Ciência da Computação) – Instituto de Computação, Universidade Estadual de Campinas.

FERNANDES, Abílio; NOVAIS, Renato. **Sistemas de Gerência de *Workflow* em *e-Science***. [2007]. Disponível em: <http://www.inf.puc-rio.br/~casanova/>. Acesso em: 5 setembro 2007.

FISHER, Paul. **A user guide to the Taverna workflow workbench**. 2007. Disponível em: <workflows.mygrid.org.uk/repository/myGrid/PaulFisher/**Taverna**_User_**Guide**.ppt>. Acesso em: 12 novembro 2007.

HINE, Christine M. **New infrastructures for knowledge production:** understanding E-science. 1.ed. United States of America: Information Science Publishing, 2006. 306p.

KEPLER Project. **Kepler Project**. 2007. Disponível em: <http://kepler-project.org/>. Acesso em: 21 novembro 2007.

KROTH, Eduardo. **Bancos de Dados Biológicos**. 2003. Disponível em: <www.inf.ufrgs.br/~clesio/cmp151/cmp15120031/BDsBiologicos.pdf>. Acesso em: 05 setembro 2007.

KROTH, Eduardo. **Integração de bases de dados**. Apresentação da disciplina de Tópicos Avançados em Bancos de Dados. Universidade de Passo Fundo. [2007].

LAUSCHNER, Tanara. **Ambientes de e-Science e a Evolução dos Padrões de Arquitetura de Computação em Grade**. Rio de Janeiro, 2005. Disponível em: <http://www-di.inf.puc-rio.br/~endler/semGSD/monografias/>. Acesso em: 30 agosto 2007.

LEMOS, Melissa. **Workflow para bioinformática**. Rio de Janeiro, 2004. 239f. Tese (Doutorado em Informática) – Programa de Pós-Graduação em Informática, Pontíficia Universidade Católica do Rio de Janeiro.

LEYMANN, Frank; ROLLER, Dieter; THATTE, Satish. **Goals of the BPEL4WS Specification**. 2007. Disponível em: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel>. Acesso em: 8 outubro 2007.

LOUSÃ, Mário; SARMENTO, Anabela; MACHADO, Altamiro. **Sistema de automatização de processos de negócios *(Workflows Systems)*:** Considerações sobre o contexto organizacional e proposta de estrutura de análise do seu impacto nas organizações. 2007. Disponível em: <www.di.fc.ul.pt/~paa/projects/conferences/coopmedia2000/lousa.pdf>. Acesso em: 05 dezembro 2007.

LUDÄSCHER, Bertram. **Scientific Workflows:** Towards a new Synthesis for Information Integration. [2007] Disponível em: <db.cis.upenn.edu/iiworkshop/postworkshop/positionPapers/130.pdf>. Acesso em: 26 setembro 2007.

LUDÄSCHER, Bertram; ALTINTAS, Ilkay; BERKLEY, Chad; HIGGINS, Dan; JAEGER, Efrat; JONES, Matthew; LEE, Edward A.; TAO, Jing; ZHAO, Yang. **Scientific Workflow Management and the KEPLER System**. 2004. Disponível em: <www.sdsc.edu/~ludaesch/Paper/kepler-swf.pdf>. Acesso em: 22 novembro 2007.

MATTOS, Amanda; MATTOSO, Marta. **Uma Estratégia para Gerência de Dados de Workflows Científicos no Contexto da Bioinformática.** In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCOS DE DADOS, 6, 2007, João Pessoa. Anais... João Pessoa: [s.n.], [2007].

MATTOSO, Marta. **Gerência Universal de Dados.** Apresentação da disciplina de Tópicos em Banco de Dados II, COPPE/UFRJ. [2007].

OASIS. **Web Services Business Process Execution Language Version 2.0**. 2007. Disponível em: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel>. Acesso em 08 outubro 2007.

OINN, Tom; STEVENS, Robert; GOBLE, Carole. **The Data Playground – data driven workflow construction for the life sciences**. 2007. Disponível em: <www.ebi.ac.uk/~tmo/dataplayground_extended.pdf>. Acesso em: 16 novembro 2007.

OLIVEIRA, Cíntia Carvalho; OLIVEIRA, Daniele Carvalho. **Bancos de Dados para Bioinformática**. [2006] Disponível em: <www.fisiocomp.ufjf.br/seminarios/BDB.pdf>. Acesso em: 5 setembro 2007.

PASTORELLO JR, Gilberto Z.; MEDEIROS, Claudia B.; SANTANCHÈ, André. **Applying Scientific Workflows to Manage Sensor Data**. In: 2007 e-Science Workshop, 2007, João Pessoa. Anais… João Pessoa:[s.n], 18 outubro 2007. 92p. p.9-18.

PENNINGTON, Deana D. **Supporting Large-Scale Science With Workflows**. 2007. Disponível em : <http://delivery.acm.org/10.1145/1280000/1273369/p45-pennington.pdf?key1=1273369&key2=0019480911&coll=GUIDE&dl=ACM&CFID=15151515&CFTOKEN=6184618>. Acesso em: 26 setembro 2007.

SILVA, Cláudio T.; FREIRE, Juliana. **Supporting Data Exploration through Visualization**. 2007. Disponível em: <http://www.vistrails.org>. Acesso em: 16 novembro 2007.

TAVERNA Project Website. **Taverna 1.6 Manual**. 2007. Disponível em: <http://taverna.sourceforge.net/index.php?doc=download.html>. Acesso em: 12 novembro 2007.

TAYLOR, Ian J.; DEELMAN, Ewa; GANNON, Dennis B.; SHIELDS, Matthew. **Workflows for e-Science: Scientific Workflows for Grids**. 2006. Springer. 1 edição. 530 páginas.

VIEIRA, Tatiana Almeida Souza Coelho; CASANOVA, Marco Antonio. **Execução Flexível de Workflows**. Rio de Janeiro. 2005. 429p. Tese de Doutorado.Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro.

VISTRAILS. **VisTrails Overview**. 2007. Disponível em: <http://www.vistrails.org/index.php/Main_Page>. Acesso em: 16 novembro 2007.

WSFL. **Web Services Flow Language**. 2007. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-ref7/>. Acesso em: 04 Novembro 2007.

XLANG. **XLANG/s Language**. 2007. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa577463.aspx>. Acesso em: 04 Novembro 2007.