UFRJ – Universidade Federal do Rio de Janeiro
COPPE – Coordenação de Programas de Pós-graduação em Engenharia
PESC – Programa de Engenharia de Sistemas de Computação

Rio de Janeiro, Brasil, Fevereiro de 2008

Gerência de Workflows Científicos: Uma Análise Crítica No Contexto da Bioinformática

Amanda Mattos
Fabio Coutinho da Silva
Nicolaas Ruberg
Sérgio Manuel Serra da Cruz

Profª. Marta Lima de Queirós Mattoso, DSc.

Índice Geral

Índice de Figuras

Índice de Tabelas

Gerência de Workflows Científicos: uma análise crítica no contexto da bioinformática

RESUMO:
Esse relatório técnico tem por objetivo apresentar as principais características de sistemas de gerência de workflows (SGWf) no apoio à experimentos científicos. SGWf vêm sendo utilizados há algum tempo no contexto de aplicações comerciais. Mais recentemente, experimentos científicos desenvolvidos em larga escala vêm necessitando de um controle sistemático, reprodutível e documentado. Assim, SGWf vem surgindo como uma solução importante, desde que devidamente adaptados ao contexto científico. Assim, esse relatório descreve os principais conceitos e requisitos dos workflows científicos, apresenta alguns sistemas disponíveis e aplicações reais em bioinformática e discute os desafios face às tecnologias existentes.

Scientific Workflow Management: a critical analysis with the bioinformatics scenario

ABSTRACT:
This technical report aims at presenting the main concepts of Workflow Management Systems (WfMS) focused on scientific experiments. WfMS have been long used in business processes. Recently, the large scale of scientific experiments is requiring a support to provide systematic control, reuse and documentation for these experiments. WfMS is been pointed as an important technology for such support when adapted for scientific requirements. Thus, this report presents the main concepts and requirements of scientific workflows, describes some available WfMS for scientific applications and shows real bioinformatics experiments while discussing the main challenges faced by current technology.

1 - Introdução

A tecnologia da informação está revolucionando a forma como a ciência do século 21 é conduzida, ela permite que grupos de pesquisadores geograficamente dispersos compartilhem dados, idéias e experimentos em tempo real. Também facilita a exposição dos resultados das pesquisas e, abrindo novas perspectivas analíticas, permite que a ciência evolua ainda mais rapidamente. Áreas multidisciplinares tais como a bioinformática, a informática médica, a quimoinformática, a ecoinformática e a geoinformática beneficiam-se diretamente dela, e em especial das áreas de algoritmos, workflows científicos, bancos de dados, sistemas de anotações, integração de dados, ontologias, processamento distribuído e inteligência artificial (Ludächer et al., 2005).

Consideramos que a Bioinformática é uma área de conhecimento convergente e multidisciplinar que envolve o intenso uso de ferramentas computacionais. Nela, os computadores são utilizados para resolver problemas nas ciências da vida; principalmente, envolve a manipulação de grandes bancos de dados de genomas, seqüências protéicas, etc. De maneira complementar, ela também envolve técnicas como a modelagem tridimensional de biomoléculas e demais sistemas biológicos (Dávila, 2006, 2007). De forma resumida, acreditamos que a Bioinformática tem como objetivos finais a coleta, a organização, o armazenamento, a recuperação e as análises de dados biológicos, propiciando a inferência ou descoberta de informações sobre a biologia e/ou evolução dos organismos.

Neste contexto, ressalta-se a existência de inúmeros projetos transnacionais de grande importância, por exemplo, o Projeto Genoma Humano (NIH, 2004). No Brasil, também existem inúmeros projetos que envolvem tanto o seqüenciamento, quanto análises genômicas e/ou proteômicas de organismos que provocam patogenias em vegetais, humanos ou animais. Neste caso, citam-se como exemplos o seqüenciamento genético da bactéria *Xylella fastidiosa* (Simpson et al. 2000); as iniciativas do consórcio RioGene (DBM, 2006) envolvido com o mapeamento e anotação das seqüências nucleotídicas das bactérias *Gluconacetobacter diazotrophicus* e com a anotação de ESTs[1] (Expressed Sequence Tags) da *Rhodnius prolixus*. Outra iniciativa de grande destaque é o projeto Genoma Eucalipto (FORESTs) que identificou 15 mil genes por meio de sequenciamento de ESTs obtidos a partir de diferentes tecidos vegetais e comparou as redundâncias com seqüências depositadas em bancos de dados públicos (Carrer, 2002). Finalmente, também é importante ressaltar os trabalhos do projeto BioWebDB (2008), relacionados não só com o sequenciamento e anotações dos genomas de vários tripanosamatídeos, como também no desenvolvimento de novas ferramentas para a área da bioinformática.

Normalmente, os projetos de bioinformática estão centrados em três grandes grupos de experimentos (*in vitro, in vivo* e *in silico*). Os experimentos científicos,

[1] EST é uma seqüência de DNA (quase sempre incompleta) que representa um *gene expresso*, ela pode ser utilizada na identificação um gene.

independentemente do domínio de aplicação, devem obedecer a um conjunto rígido de requisitos (Weske, Vossen, Medeiros, 1996).

1. Os experimentos devem ser reprodutíveis e disseminados, permitindo que outros cientistas conduzam experimentos semelhantes que confirmem (ou refutem) os resultados. Os resultados devem ser documentados e podem ser usados com base para futuras pesquisas.
2. Os experimentos devem seguir um protocolo, também conhecido como metodologia. Eles podem ser conduzidos a partir de um ponto inicial, ou mais comumente, eles podem reutilizar ou combinar experimentos (etapas) realizados anteriormente.
3. Os experimentos devem ser executados em condições controladas. Geralmente, um experimento é uma composição ordenada de etapas (complexas). As condições experimentais deverão ser documentadas e controladas ao longo das etapas, assim, caso seja necessário refazer uma etapa, os mesmos resultados serão obtidos.

O objetivo desse relatório técnico está ligado ao terceiro grupo de experimentos, também conhecidos como experimentos *in silico*. Os experimentos *in silico* fazem intensos usos de várias ferramentas computacionais. Elas podem ser utilizadas de forma encadeada, que vão desde a construção de cromatogramas, passando pela montagem de seqüências e suas respectivas correções, análises de alinhamentos, comparações de seqüências, visualizações de dados e, finalizando com as inclusões de anotações (meta-informações) sobre uma dada seqüência.

Esse perfil operacional leva ao crescente aumento dos dados produzidos durante a execução do experimento. Consequentemente, o vertiginoso crescimento dos dados ligados a Biologia Molecular traz diferentes possibilidades de análises, mas também novos níveis de dificuldade para os pesquisadores para a compreensão, integração e manipulação desses dados (Galperin, 2007). Dentre os grandes desafios, destacam-se: Os dados possuem diferentes origens, versões e nem sempre são confiáveis; os dados podem ser armazenados em diferentes formatos e locais; os dados apresentam diferentes graus de qualidade e confiabilidade; os dados são armazenados em repositórios que possuem estruturas heterogêneas e métodos de acesso distintos. Todas essas características, aliadas ao elevado número de ferramentas computacionais, muitas vezes levam o pesquisador a atuar como um "informata", deslocando-o de seu objetivo inicial.

Este relatório está dividido da seguinte forma. Na seção 2 apresentamos os principais conceitos relacionados com os workflows científicos, a seção 3 apresenta as principais características (semelhanças e diferenças) entre os workflows comerciais e científicos. Na seção 4 apresentamos os principais métodos de representação formal de workflows e discutimos suas principais características. A seção 5 descreve, duas ferramentas para gerência de workflows científicos (Kepler e myGrid), além disso apresentamos as principais tecnologias que podem ser associadas. Na sexta seção apresentamos exemplos reais de workflows científicos. A ultima seção apresenta a conclusão deste trabalho.

2 - Descrição e conceitos

Esta seção apresenta os principais conceitos relacionados aos workflows e seus sistemas de gerenciamento.

2.1 - Workflows

Das muitas tarefas desenvolvidas pelos pesquisadores, algumas dizem respeito à composição (seqüência) de programas de bioinformática, onde cada ente produz uma coleção de dados com determinada semântica e sintaxe. Essa coleção poderá ser utilizada como entrada de dados para o próximo programa. Ressalta-se que a composição de programas não é uma tarefa trivial e, em muitos casos torna-se uma barreira para análises mais sofisticadas. Na computação, o arcabouço funcional que permite a composição de programas em uma seqüência de execução com o objetivo de gerar um resultado final é chamado de *workflow* (Wf).

O termo workflow não é uma exclusividade do ambiente científico, pelo contrário ele tem suas origens associada ao processo de automação de escritórios, segundo Araújo e Borges (2001) essa tecnologia originou-se por volta dos anos 1970. O foco dessa tecnologia era oferecer soluções voltadas para a geração, armazenamento, compartilhamento e roteamento de documentos em uma organização, com o objetivo de reduzir a manipulação (física) de documentos em papel. Com o passar do tempo percebeu-se que esse foco deslocou-se para a gerência e automação de processos de negócio[2].

Um workflow provê a abstração necessária para descrever uma série de processos estruturados e suas atividades com o fito de prover um ambiente robusto de resolução de problemas e assim, promover o uso efetivo e otimizado dos recursos computacionais (Hollingsworth, 1995). Apesar dos workflows já serem estudados durante muito tempo, existem na literatura diversas definições e classificações de workflows. A definição, segundo WfMC (1999), é:

> *"A automação de um processo de negócio, completo ou apenas parte dele, através do qual, documentos, informações ou tarefas são transmitidos de um participante a outro por ações, de acordo com regras procedimentais."*

Percebe-se que esta definição não representa de maneira ideal os workflows científicos. Uma definição mais simples e mais geral é apresentada por Moro (2001) como "Qualquer tarefa executada em série ou em paralelo por dois ou mais membros de um grupo de trabalho visando um objetivo comum". Nesta definição é possível atingir um largo campo de atividades, tanto comerciais quanto científicas.

Alguns autores utilizam essa tecnologia com o foco na área de trabalho cooperativo, enquanto outros a utilizam com o foco na automação de tarefas. Mesmo assim, ainda

[2] Em tempo, processos de negócio podem ser compreendidos como um procedimento onde documentos, informações e tarefas são passadas entre os membros de uma ou mais organizações, segundo um conjunto de regras pré-definidas, e que devem ser realizadas para alcançar o objetivo do negócio (Cruz,2000)

existem outras classificações. Por exemplo, para Georgakopoulos et al (1995) e Plesums (2002), os workflows comerciais podem ser classificados em três tipos básicos: *ad-hoc*, administrativo e de produção. Entretanto, Weske, Vossen, Medeiros (1996) e Singh e Vouk fluxo de procedimentos e tarefas, onde cada programa participante opera segundo um conjunto de regras definidas *a priori*, ou seja, ele receberá um conjunto de dados, os processará e os enviará para o próximo programa.

2.2 - Workflows Científicos

Apesar de os workflows terem inicialmente surgido no ambiente de negócios (Cruz, 2000, Aaslt, Hee 2002), cada vez mais, a Biologia e demais ciências ligadas à Matemática e à Natureza (Química, Física, Geografia, Engenharia, entre outras) utilizam-nos. Na bioinformática, existem diversos tipos de workflows, muitos produzidos através do uso de linguagens de *scripts*. Esses *scripts* definem em um só momento não só a seqüência (completa) de execução, como também os parâmetros de execução do fluxo de trabalho. No entanto, ressalta-se que o uso de *scripts* pode causar inúmeros transtornos para os pesquisadores, como por exemplo, a grande especificidade de um *script* pode dificultar sua manutenção e reduzir a capacidade de reutilização. Além disso, existem outras deficiências, tais como as dificuldades do registro: das execuções dos programas; da origem dos dados utilizados; das transformações aplicadas aos dados; dos resultados obtidos, entre outras. Na tabela 2.1, encontram-se as principais características dos workflows científicos apresentadas por Weske, Vossen, Medeiros (1996), Altintas et al. (2006), Ludäscher, Altintas (2006) e Sangeeta (2005).

Característica	Descrição
Interface	O "*design*" do workflow deverá ser intuitivo e o ambiente de execução voltado para o usuário final, com isso espera-se que detalhes de implementação (baixo nível) sejam escondidos, permitindo que o pesquisador concentre-se apenas do nível conceitual do workflow
Reutilização	Os componentes deverão ser reutilizáveis e intercambiáveis, em especial, os workflows devem ter capacidade de adicionar/remover novos processos dinamicamente
Transformação de dados	Deverá ser flexível o bastante para permitir consecutivas transformações de dados entre as atividades de um processo
Interações e processamento em lote	Deverá suportar "*process steering*" (*play, pause* e *stop*) durante a execução de um processo. Deverá ter habilidade de monitorar um processo em tempo de execução mesmo que ele esteja sendo executado em segundo plano
***Streaming* (execução parcial)**	Deverá suportar *streams* de dados (de moderada a intensa)
Localidade	Deverá suportar processamento local e distribuído do workflow
Interoperabilidade	Deverá suportar alterações na descrição do workflow (coleções de dados e sistemas de análise)
Complexidade	Deverá ser capaz de manipular complexos fluxos de dados, controles e eventos
Desempenho e planejamento	Deverá ser capaz de informar o desempenho e os custos de execução. Também deverá ser capaz de coletar dados de diferentes processos e fazer uso de métricas com o objetivo de predizer quais serão os tempos de execução
Dependência	Deverá estar associado a um sistema de gerência confiável, de alta disponibilidade e tolerante a falhas
Verificação e Validação	Deverá verificar e validar a construção ou importação de workflows
Alterações dinâmicas	A definição do workflow é uma atividade dinâmica, é influenciada pelos resultados intermediários, logo, o workflow poderá sofrer alterações em seus fluxos de execução
Modelagem e simulação	Devem suportar etapas que permitam aos cientistas simular e modelar condições experimentais

Tabela 2.1 – Principais caracterísitcas dos workflows científicos

Os workflows científicos guardam uma correlação muito estreita com os experimentos científicos propriamente ditos, em algumas vezes podem ser até mesmo confundidos uns com os outros. Os experimentos podem ser executados inúmeras vezes com ligeiras modificações nos programas ou em parâmetros operacionais, todo esse processo, independentemente da natureza do workflow é interativo e incremental, portanto, alterações na definição conceitual de um workflow científico são constantes.

Os resultados produzidos pelos workflows científicos devem ser armazenados, pois são uma fonte de dados imprescindível para os cientistas. Grosseiramente, os resultados podem ser positivos ou negativos. Os primeiros podem corroborar as hipóteses postuladas no início do experimento enquanto que os demais, também possuem grande importância, eles podem indicar erros experimentais ou mesmo erros na execução total ou parcial de um workflow. Além do armazenamento dos resultados, é importante manter o registro do processo de obtenção dos dados, assim é possível garantir a

reprodutibilidade de um dado experimento ou ainda o reuso dos dados em outros experimentos. Essa característica a cada dia se torna mais importante, pois além de garantir economia de recursos (computacionais e humanos) também otimiza o tempo dos cientistas. O mecanismo de registro dos resultados, das versões de programas e dos workflows propriamente dito chama-se proveniência, ele será apresentado e discutido em detalhes nas próximas seções.

O processo de definição de um workflow científico não é regido por metodologias nem é independente de domínio de aplicação, pelo contrário, cada workflow requer uma ampla discussão entre os membros do grupo de pesquisa, sua definição geralmente envolve tomada de decisão e análises refinadas sobre cada uma das etapas. Entretanto, apesar dos workflows científicos possuírem características particulares, os cientistas desejam aumentar a eficácia e a eficiência nos experimentos científicos, portanto, faz-se necessário o uso de sistemas de gerência de workflows (SGWf). Suas principais características serão discutidas nas próximas seções.

2.3 - Sistemas de Gerência de Workflows

Segundo a WfMC (*Workflow Management Coallition*), organização fundada em 1993 por vendedores, fabricantes e usuários de Workflows com o objetivo de padronizar os conceitos e as tecnologia de workflow, é possível definir um SGWf como (WfMC, 2001):

> "*Sistemas para a definição, criação e gerência da execução de fluxos de trabalho através do uso de software, capazes de interpretar a definição de processos, interagir com seus participantes e, quando necessário invocar ferramentas e aplicações.*"

Os SGWf são pacotes de software que fornecem toda a infra-estrutura para definir, executar e monitorar workflows. A utilização desse tipo de ferramenta é importante para separar a definição do workflow, do mecanismo (*engine*) encarregado de executá-lo, permitindo dessa forma que modificações em uma das partes não afetem o funcionamento da outra. Neste sentido, o WfMC preconiza que os SGWf devem apresentar as características genéricas representas da Figura 2.1.

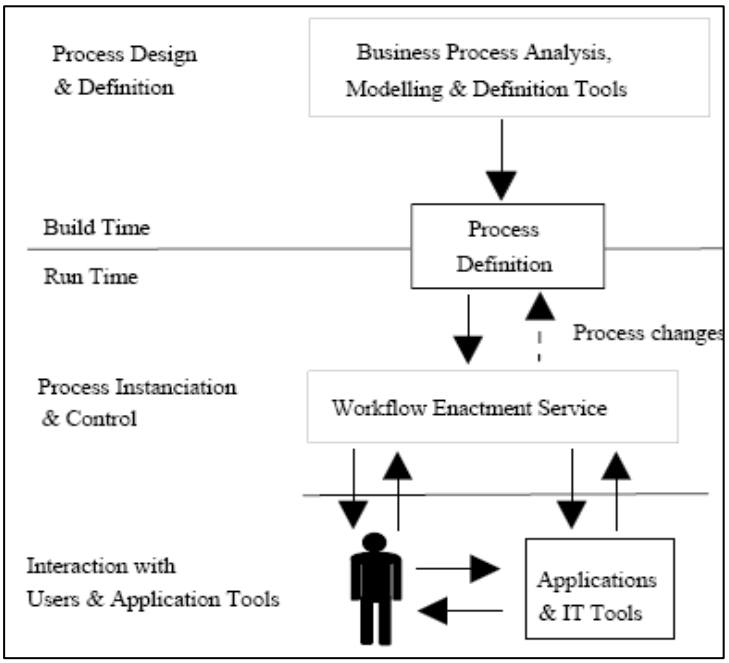

Figura 2.1 - Estrutura genérica de um SGWf (WfMC, 2001)

Os SGWfs podem ser subdivididos em duas seções bem distintas. Apesar de complementares, elas atuam em momentos distintos. A primeira (*build time*) permite que um projetista trate das questões relacionadas com a análise e a modelagem dos processos que serão executados *à posteriori*. Nesse primeiro momento o projetista define os processos e seus planos de execução com elevado níveis de abstração. A seguir, essas definições são transformadas, através de compilações ou qualquer outro mecanismo de tradução para uma linguagem intermediária que será executada pela máquina de workflow[3].

A máquina de workflow, aqui denominada *Workflow Enactment Service*, atua na instanciação, roteamento e controle dos processos submetidos à execução. Nesse momento (*run time*) ela tanto pode interagir com ferramentas e aplicações externas, quanto com o projetista ou outros usuários que porventura tenham interesses em acompanhar uma instância qualquer de um processo ou ainda monitorar o funcionamento da máquina como um todo. Ao refinarmos um pouco mais a análise sobre o padrão da WfMC, percebe-se que os SGWf são, geralmente, construídos segundo o paradigma cliente-servidor. Na porção cliente, se encontram as aplicações que habilitam a definição dos processos, a modelagem de workflows, as interações e, em especial, as funções de administração e monitoramento do sistema. Na porção servidora, tem-se a máquina de workflow propriamente dita, atuando na execução dos processos.

Com o objetivo de ampliar a interoperabilidade entre as aplicações e ferramentas e padronizar a terminologia e a representação da modelagem de processos o WfMC definiu um modelo de referência de workflow, denominado (*Workflow Reference Model*) (WfMC, 2001). Esse modelo define um conjunto de cinco interfaces e

[3] Alguns autores também chamam a máquina de execução de workflow de ambiente de execução de workflow ou ambiente de encenação de processos (Araújo, Borges, 2001).

protocolos para a comunicação. A estratégia adotada pelo WfMC foi muito interessante, ela estabeleceu uma fronteira em torno do ambiente de execução de workflow e padronizou vários atributos funcionais através da adoção de um conjunto de APIs. Assim, ela limitou-se a definir "o que" se poderia fazer com as máquinas de workflow, deixando a cargo dos fabricantes "como" fazê-lo.

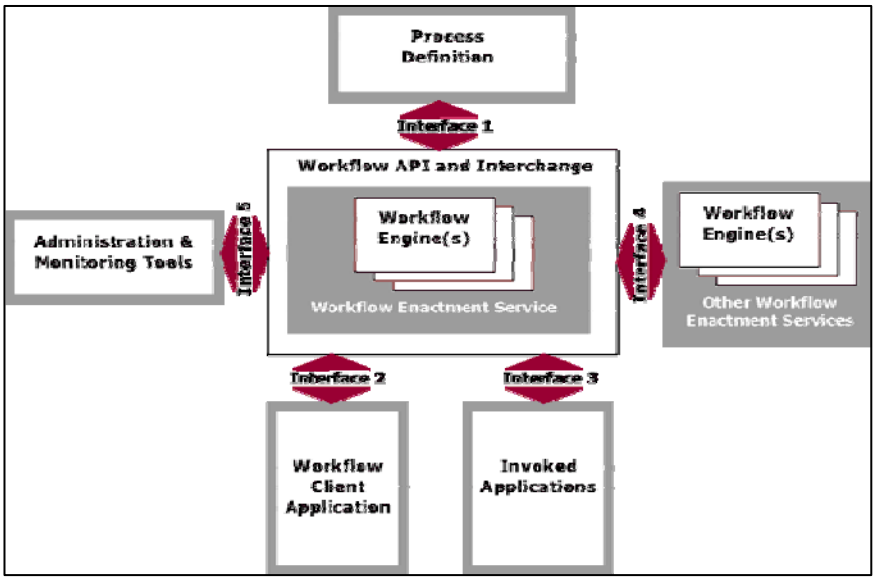

Figura 2.2 – Modelo de referência de workflow (WfMC, 2001)

Cada uma das cinco interfaces interage com um conjunto de serviços externos ao ambiente de execução (Figura 2.2). Portanto, as aplicações de workflow são capazes de apresentar diferentes graus de conformidade e integração ao modelo proposto. Na tabela 2.2, encontram-se as principais características de cada interface do modelo.

Interface	Nome da Interface	Descrição
Interface 1	Descrição de processo	Possibilita a escolha de diversas ferramentas de modelagem de processos distintos. Baseia-se em uma interface que permite a importação e exportação de processos
Interface 2	Aplicações cliente	Constrói uma lista de trabalhos comuns e prove gerenciamento único para vários sistemas de workflow independentemente do software utilizado no SGWf. Essa faceta permite a oferta de serviços de sistemas distintos combinados em uma mesma interface, deixando transparecer que o usuário está lidando com um única máquina de workflow
Interface 3	Integração com ferramentas e aplicações externas	Permite que aplicações ou qualquer ferramenta externa seja utilizada de forma padronizada, adicionalmente, ela também possibilita o desenvolvimento de agentes de aplicações padronizados.
Interface 4	Interoperabilidade entre máquinas de workflow	Suporta o desenvolvimento de aplicações de automação de processos de workflow usando diferentes máquinas de execução. Além disso, ela também possibilita que aplicações heterogêneas de workflow compartilhem dados ou mesmo outras aplicações.
Interface 5	Administração e ferramentas de monitoramento	Suporta funções comuns de gerenciamento, administração e auditoria.

Tabela 2.2 – Interfaces do Modelo de referencia do WfMC

2.3.1 - Classificação de Sistemas de Gerência de Workflows

A classificação de SGWf não é uma tarefa trivial. Inúmeras tentativas de classificação de sistemas de gerenciamento de workflows já foram propostas, entretanto ainda não há um consenso. Por exemplo, WARIA (2004) afirma que parecem existir tantas classificações quanto o número de fabricantes de sistemas de workflow. Araújo e Borges (2001) apresentam uma classificação que leva em consideração as abordagens de comunicação. Marshack, (1995) classifica os workflows de acordo com o grau de estruturação dos processos. Por fim, uma terceira forma, talvez a mais polêmica de todas, tenta classificar os SGWf de acordo com o seu uso (comerciais, científicos, grid, entre outros). Apesar de ser uma forma de classificação muito fluida, ela é muito utilizada por inúmeros autores ligados à área de Bioinformática. Nesse contexto, muitos autores classificam os SGWf de modo simplório. As principais classificações são: SGWf comerciais, científicos (Weske, Vossen, Medeiros, 1996), (Singh, Vouk,1998) e (Lemos, 2004) e SGWfs em grid (Yu, Buyya, 2005), (Spooner et al 2005). Os dois últimos grupos são objetos de estudo desse relatório técnico e serão apresentados nas próximas seções.

2.3.1.1 - Classificação quanto ao tipo de comunicação

Este tipo de classificação leva em consideração a comunicação entre os atores dos processos e o sistema de workflow, eles podem ser classificados como: sistemas de compartilhamento de documentos e sistemas baseados em mensagens (Araújo e Borges, 2001).

Nos sistemas de compartilhamento, os documentos que são manipulados ao longo do processo, sendo armazenados em uma base de dados compartilhada. Por intermédio de listas de trabalho individuais, os atores são instruídos sobre quais documentos utilizar para a execução de uma determinada tarefa e o sistema de workflow se encarrega de oferecer os acessos necessários à base de dados. Nos sistemas baseados em mensagens os atores e a máquina de workflow solicitam a execução de atividades através da emissão de mensagens. Além da solicitação da execução de uma determinada tarefa, as mensagens podem conter os documentos necessários à realização da tarefa.

2.3.1.2 - Classificação quanto ao grau de estruturação de processos

Segundo Marshack (1995), os SGWf são divididos em três categorias (*ad-hoc, administrativos e produção*). Essa divisão baseia-se tanto no grau de estruturação dos processos quando na freqüência de repetição dos fluxos de trabalho que os apóiam.

Sistemas de workflows ad-hoc são aqueles cujos workflows são fracamente estruturados. Os processos e seus encadeamentos são quase sempre pouco conhecidos ou mesmo imprevisíveis até o momento da execução propriamente dita. Esse tipo de sistema é voltado para grupos dinâmicos que executam processos únicos e altamente individualizados. Normalmente, neste tipo de aplicação, os usuários finais tornam-se ao mesmo tempo os desenvolvedores do workflow e gerentes de seus próprios processos. É importante ressaltar que alguns autores, principalmente aqueles relacionados com a área de trabalho cooperativa, também classificam este tipo de sistema como sistemas de workflow colaborativo. Em geral, estes sistemas são utilizados por equipes em trabalhos que envolvam a produção de conhecimento, isto é, trabalhos que requerem criatividade

e flexibilidade. Exemplos de processos *ad-hoc* seriam a construção de relatórios por equipe, a organização de um evento ou mesmo a definição de um projeto de sistema. Apesar de não encontrar referências sobre workflows *ad-hoc* na bioinformática, acreditamos que este tipo de workflow se adapta muito ao contexto dos experimentos *in silico*, uma vez que tais experimentos sofrem freqüentes atualizações, motivadas por uma visão mais refinada do problema ou mesmo por mudanças de etapas e tarefas que deixaram de ser relevantes para ao processo de exploração científica.

Sistemas de workflow do tipo *administrativo* são capazes de gerenciar processos mediamente estruturados. Neste caso há maior previsibilidade no encadeamento das tarefas e um mesmo workflow pode ser executado inúmeras vezes sem grandes alterações nos fluxos. Com relação aos usuários do sistema é comum que todos os membros da organização sejam usuários, portanto, as aplicações que suportam este tipo de workflow deverão ter alta escalabilidade. Esta participação, todavia, não é obrigatória, pelo contrário, é ocasional, a execução de processo não é a única atividade realizada pelo usuário dentro da organização. Exemplos de processos administrativos são: a inscrição em disciplinas em uma universidade ou preenchimento de pedidos de compra em uma organização. Apesar da nossa revisão de literatura não ter detectado nenhum trabalho que correlacione sistemas de workflow administrativo à bioinformática, não é difícil traçar um paralelo entre os dois enfoques principalmente se considerarmos que os processos na bioinformática são estruturados e reprodutíveis e ainda, nem todos os cientistas de um grupo de pesquisa trabalham exclusivamente com a execução de workflows.

Sistemas de workflow de *produção* são indicados para tarefas que seguem uma estruturação rígida e bem definida. As regras e o encadeamento dos processos são bem conhecidos e facilmente determinados através da análise do processo corrente. Geralmente, esses processos têm elevada freqüência de repetição e a sua execução é a principal atividade dos seus usuários de um determinado setor. Os participantes deste processo também podem realizar outras tarefas, entretanto o processo apoiado pelo workflow é o mais importante. Por exemplo, o atendimento a clientes em uma *helpdesk*, onde os consultores passam a maior parte do tempo executando o processo de tele-atendimento.

As categorias aqui representadas não devem ser visualizadas de forma estanque ou mutuamente exclusivas uma vez que os processos podem apresentar ligeiras variações. Os processos raramente são puramente *ad-hoc*, ou exclusivamente *administrativos* ou totalmente de *produção*. Chaffey (1998) corrobora e ainda amplia essa observação. Ele sugere uma classificação ainda mais abrangente, isto é, ele não só leva em conta o grau de estruturação dos processos apoiados pela aplicação, como também adiciona uma nova dimensão em relação à necessidade de colaboração requerida entre os atores para a execução do workflow. Acreditamos que os sistemas de workflow utilizados na área de bioinformática também não podem ser classificados de forma rígida, uma vez que muitos destes sistemas evoluíram junto com computação, cujas fronteiras são tênues e altamente mutáveis. Em muitos casos, as aplicações científicas utilizam combinações de seqüências de recursos distintos, tais como aplicações locais e serviços Web remotos, bancos de dados públicos, transformações de dados, entre outros.
Nas próximas seções apresentaremos as principais características dos SGWf comerciais e científicos.

2.3.1.3 - Sistemas de Gerência de Workflows Comerciais

Mesmo nesta categoria de sistemas não há uma definição muito clara. Segundo Aalst (2004) os primeiros SGWfs comerciais surgiram nos anos 80. Atualmente existem diversos SGWfs comerciais (figura 2.3), como por exemplo, Staffware da Staffware Plc, o COSA da Ley GmbH, o ActionWorkflow da Action Technologies Inc., o MS Message Queuing (MSMQ) da Microsoft, o Oracle Workflow da Oracle, o Lotus Workflow da IBM, entre tantos outros. Esses softwares fornecem desde mecanismos para o suporte básico de workflows até soluções completas para a definição, execução e gerenciamento. Esses produtos suportam a especificação (modelagem) de workflows. Essa especificação pode ser feita através de linguagens de definição (abertas ou proprietárias), utilizando diagramas, grafos, descrições textuais ou até mesmo documentos XML (Santos, 2004).

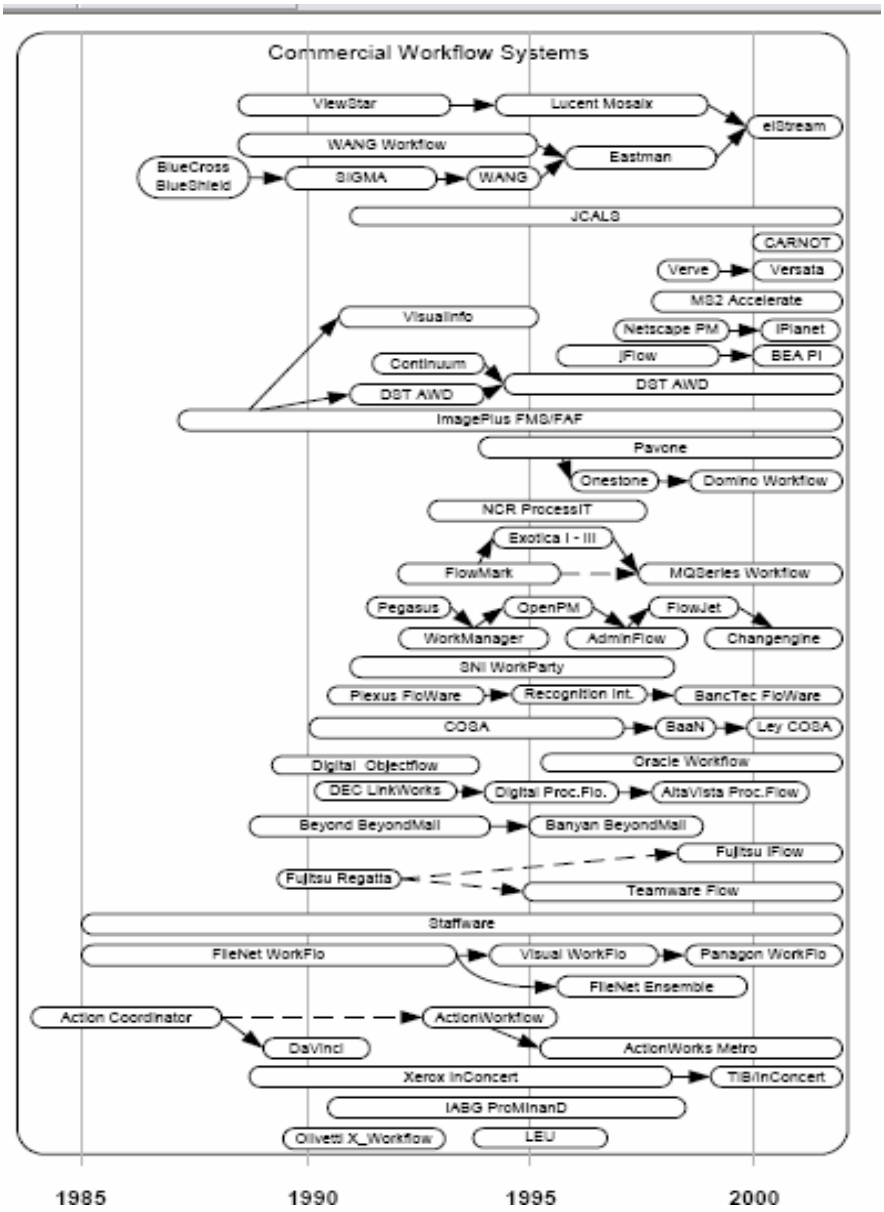

Figura 2.3 - Perspectiva histórica dos principais SGWfs comerciais ao longo do tempo (Aalst, Dumas, Hofstede , 2004)

Na literatura, é possível encontrar diversas linguagens de definição de workflows, todavia ainda não há um consenso sobre uma linguagem padrão. Existem inúmeras propostas diferentes como ebXML, WSFL, XLANG e BPML. Recentemente, a IBM, a Microsoft e a BEA lançaram a linguagem BPEL4WS (BPEL) (*Business Process Execution Language for Web Services*) (Curbera, Goland, Andrews, 2003).

A linguagem BPEL possui um rico, medianamente formal, conjunto de construtores XML utilizados para coordenar processos de negócio encapsulados em serviços Web. BPEL é um esforço comum entre grandes fornecedores do mercado, ela se propõe a substituir as antigas propostas da IBM (WSFL) e Microsoft (XLANG). Portanto, muitos autores acreditam que ela é um sério candidato a se tornar a linguagem padrão para a definição de workflows que utilizam serviços Web no ambiente comercial. No entanto, Aalst, Hee (2002) e Aalst, Dumas, Hofstede (2004) advogam que apesar de ser uma boa linguagem de composição, existe um certo afrouxamento na sua definição formal. Os autores sugerem o emprego de redes de petri como fundamentação teórica. Já Braghetto et al. (2005) sugerem o uso da álgebra de processos como formalização das linguagens. As discussões sobre a necessidade de formalismos nas linguagens de composições serão apresentadas nos próximos tópicos deste trabalho.

Os SGWf comerciais têm como principal característica o foco nos fluxos de controle, sendo capazes de manipular processos, dados e recursos. Eles definem, automatizam e gerenciam uma seqüência de atividades de um fluxo de trabalho, além disso, eles podem controlar a invocação dos recursos necessários para a execução das atividades. Tradicionalmente, os SGWf devem obedecer a uma lista de requisitos (Lemos, 2004), (Gertz, Ludascher 2006) a saber:

1. Oferecer mecanismos de extensibilidade para acomodar novos processos, dados e recursos.
2. Permitir a definição e redefinição do workflow. O sistema deve permitir a definição de propriedades dos processos, dados e recursos de tal forma a facilitar a escolha dos mais adequados para cada situação-problema.
3. Validar o workflow através da verificação das entradas, saídas e conversões de dados.
4. Otimizar e executar o workflow definido pelo usuário, adicionalmente, ele poderá monitorar e intervir na execução do fluxo de trabalho.
5. Agendar a execução do workflow.
6. Armazenar automaticamente meta-descritores adicionados pelos usuários ou mesmo produzidos durante a execução dos fluxos de trabalho.

Estes requisitos também são válidos para os SGWf científicos, no entanto, existe uma dicotomia entre os dois tipos de SGWfs. Os SGWfs científicos exigem/apresentam algumas características peculiares (motivadas pela natureza dos experimentos científicos), elas raramente estão disponíveis nos ambiente comerciais. Essas características serão exploradas na próxima seção.

2.3.1.4 - Sistemas de Gerência de Workflows Científicos

O termo SGWf científico é utilizado para descrever o processo de workflow voltado para as aplicações científicas de quaisquer áreas de pesquisa, como por exemplo, biologia, física, química, ecologia, geologia, astronomia, entre outras. Essas aplicações

compartilham uma base comum, todas são baseadas em experimentos científicos que requerem alto poder computacional. Os SGWf científicos são responsáveis por invocar as aplicações (locais ou externas) que participam de um workflow científico, passando os dados pertinentes entre elas. Geralmente eles oferecem para os usuários interfaces gráficas que permitem não só a definição conceitual do workflow, como também permite o monitoramento de sua execução.

Para Lemos (2005), um SGWf científico apresenta uma lista de requisitos, dentre eles destacamos:

1. **Processos, Dados e Recursos** - O SGWf deve incluir os processos, dados e recursos normalmente usados e oferecer mecanismos de extensibilidade para acomodar novos processos, dados e recursos. Entre os processos estão os programas de bioinformática, programas (locais ou remotos) que filtram/transformam os resultados de outros programas, além de mecanismos de controle de execução do workflow.

2. **Definição** - O SGWf deve auxiliar os cientistas na definição e redefinição do workflow. A redefinição é um passo importante quando os resultados finais não forem considerados úteis ou interessantes pelos pesquisadores. Para tanto, o sistema deve permitir a definição de propriedades dos processos, dados e recursos de tal forma a facilitar a escolha dos mais adequados para cada caso exposto pelo pesquisador.

3. **Validação** - O SGWf deve oferecer ferramentas de validação do workflow definido pelo cientista. Durante a validação, o sistema deve verificar se as entradas e saídas definidas pelos usuários para cada processo do workflow são consistentes, além de incluir, caso seja necessário, processos que façam conversão nos formatos dos dados e processos que verifiquem se os resultados gerados pelos programas são esperados ou não. Este requisito é importante porque os programas de sequenciamento de bioinformática podem gerar resultados em formatos diferentes do esperado, por exemplo, a saída de um programa não é FASTA e a entrada do próximo requer o formato FASTA, isso possivelmente, implicará no não funcionamento de outros programas do workflow.

4. **Otimização, Monitoramento e Execução** - O SGWf científico 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 e deve permitir a intervenção do pesquisador em qualquer ponto. A intervenção (interrupção temporária) é necessária caso ele queira avaliar os resultados intermediários para decidir se continua ou não a execução, ou para fazer alguma modificação na definição das próximas atividades do workflow.

5. **Agendamento** - O sistema deve oferecer agendamento da execução do workflow. O cientista ou membros de sua equipe podem desejar executar o workflow uma única vez em um determinado dia ou de tempos em tempos. Por exemplo, semanalmente a equipe deseja verificar se existem novas seqüências nos bancos de dados públicos, caso afirmativo, novos alinhamentos serão produzidos.

6. **Dados e Metadados** - O SGWf científico deve armazenar tanto os dados produzidos pelo workflow quanto os metadados. Metadado é comumente definido como "dado sobre dado", ou ainda, de forma mais completa, como uma informação sobre o dado que permite o acesso e gerenciamento deste dado de maneira eficiente e inteligente (Sumpter, 1994). O sistema deve ser capaz de gerar estes metadados automaticamente e, sempre que possível, oferecer subsídios para que o cientista consulte-os e atualize-os.

,McPhillips e Bowers (2005), McPhillips 2006 e Bowers et al (2006a e 2006b), aprofundaram a discussão sobre as características dos SGWf científicos, em especial daqueles relacionados à bioinformática e filogenética[4]. Os autores definem requisitos que devem ser suportados pelos para SGWf científicos:

1. Os SGWf científicos devem suportar operações aninhadas - os *datasets* sobre filogenética e bioinformática em geral, são grandes e complexos e, em muitos casos, aninhados. Os sistemas devem operar e oferecer suporte para o processamento eficiente sobre operações aninhadas que utilizam esses *datasets*.

2. Os resultados do processamento devem ser reprodutíveis - os pesquisadores devem ser capazes de reproduzir (de modo fácil e confiável) os resultados produzidos por outros pesquisadores. O SGWf deverá registrar automaticamente os resultados. A proveniência dos dados de entrada e dados intermediários deve ser associada com os dados de saída produzidos durante o processamento de um workflow.

3. Os SGWf devem suportar o reuso de workflows - os pesquisadores devem ser capazes de desenvolver workflows genéricos ou ainda reutilizar partes de outros workflows.

4. Os SGWf devem ser robustos - eles devem ser capazes de manipular exceções causadas por dados ou parâmetros incorretos, sem causar interrupção no fluxo de execução do workflow.

2.3.1.5 - Sistemas de Workflow em Ambientes de Grid (SGWfG)

Os grids computacionais podem ser definidos como uma infra-estrutura global de computação distribuída que integra e permite o compartilhamento (ordenado, seguro e flexível) de recursos heterogêneos através de redes de computadores (Foster e Kesselman, 1999). Os grids têm como foco principal a execução de aplicações que demandam elevado poder computacional, orientadas à solução de problemas de grande complexidade, eles podem ser utilizados tanto para aplicações comerciais quanto para aplicações científicas.

Os grids permitem que as mais variadas comunidades científicas (físicos, químicos, biólogos, geólogos, entre outros) compartilhem e processem grandes conjuntos de dados, aplicações, recursos computacionais e instrumentos em geral. Entretanto, para que o compartilhamento seja proveitoso, ele deverá ser executado através sistemas de

[4] Filogenética é a área da Biologia que estuda a evolução das espécies, segundo as suas transformações.

workflow em ambientes de grid. Atualmente existem inúmeras pesquisas nessa área, eles vão desde a especificação do workflow até sua execução, passando pela orquestração, simulação e escalonamento (Spooner et al., 2005).

Para Yu e Buyya (2005), os workflows em ambientes de grid podem ser vistos como uma coleção de tarefas que são processadas em recursos distribuídos sob uma ordem bem definidas e com um objetivo definido. Para os autores os sistemas de workflow em Grid seguem a arquitetura apresentada na figura 2.4. As funcionalidades são suportadas por vários componentes que estão baseados no modelo de referência proposto pelo WfMC (WfMC, 1999).

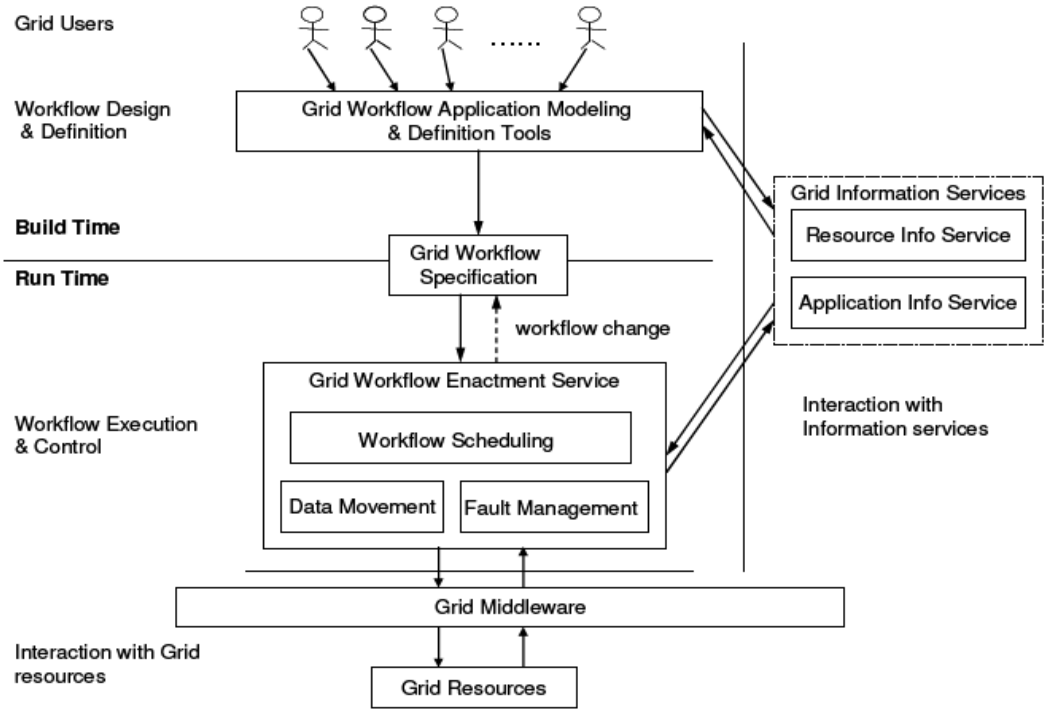

Figura 2.4 - Sistema Gerenciador de Workflows em ambiente de grid (Yu, Buyya, 2005)

Resumidamente, as principais funções podem ser classificadas como *build time* e *run time*. As primeiras estão relacionadas com a definição e modelagem das tarefas do workflow e suas dependências. As demais funções são relacionadas com o gerenciamento da execução do workflow e as interações com os recursos do Grid. Os usuários interagem como ferramentas de modelagem de workflows e produzem especificações de workflows, que são submetidas ao serviço de *enactment* do workflow. Este serviço está localizado acima dos *middlewares* de Grid (GLOBUS, 2008), UNICORE (UNICORE, 2008), Alchemi (ALCHEMI, 2008) e, é responsável pelo escalonamento, gerenciamento de falhas e movimentação de dados.

Atualmente, existem inúmeros SGWfGs, porém cada um deles possui características próprias e muitas vezes conflitantes, o que dificulta a compreensão e avaliação das suas funcionalidades. Dentre os principais SGWfG destacamos: Pegasus, GridFlow. ICENI, TITAN, GridAnt, Triana, GridBus, GryPhyN, entre tantos outros. Com vistas a reduzir esta limitação Yu e Buyya (2005) definiram uma taxonomia para classificar e caracterizar os SGWfGs (Figura 2.5).

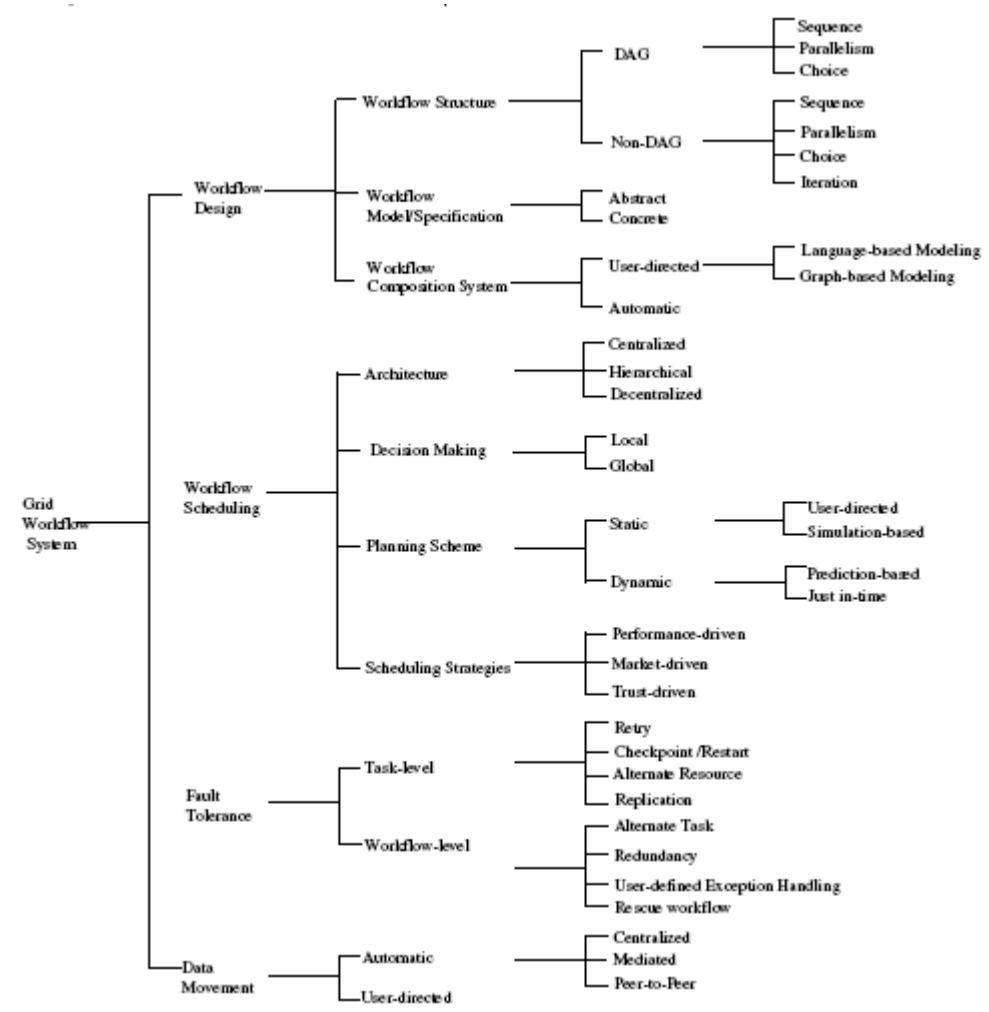

Figura 2.5 - Uma taxonomia de sistemas gerenciadores de workflows científicos em ambientes de grid. (Yu, Buyya, 2005)

A taxonomia está dividida em quatro elementos principais, suas principais características estão representadas na tabela 3.

Característica	Descrição
Design	Essa categoria define como os componentes de workflow podem ser definidos ou compostos. Ela está subdividida em três categorias. A primeira descreve as possíveis estruturas e padrões de workflow, isto é a dependência (relação temporal) entre as tarefas conectadas. As estruturas podem ser do tipo DAG ou não DAG. A segunda define as especificações do workflow. A terceira define como os usuários irão associar os componentes dos workflows
Recuperação da Informação	Nesta categoria, os SGWfG não executam as tarefas propriamente ditas, eles apenas gerenciam a sua execução e recuperam as informações produdizas de três formas distintas: *IR estática* (fornece informações que não variam ao longo do tempo, geralmente relacionadas a infra-estrutura), *IR Histórica* (fornece informações sobre eventos prévios de Wfs outrora executados) e *IR dinâmica* (fornece informações sobre o estado do Grid e recursos de CPU, memória, rede, etc.)
Escalonamento	Mantém informações sobre o escalonamento de tarefas no Grid
Tolerância à falhas	Os SGWfG devem ser capazes de identificar e controlar falhas e suportar execuções confiáveis na presença de concorrência e falhas, existem dois níveis de tolerância: *Task Level* e *Workflow Level*. O primeiro é frequentemente utilizado em sistemas paralelos e distribuídos, ele está relacionado às falhas dos componentes do workflow. O segundo requer técnicas aplicadas ao nível do fluxo de execução, usadas para tratar condições de erro do workflow.
Movimentação de dados	Esta categoria descreve como as tarefas do Grid podem movimentar os dados requeridos ou produzidos. As movimentações podem ser manuais ou automáticas, essa por seu turno podem ser centralizadas, mediadas ou ponto a ponto.

Tabela 2.3 – Taxonomia de Yu e Buyya para SGWfG

Apesar do pioneirismo do trabalho, ainda é possível encontrar uma série de conflitos entre as definições conceitos relacionados com o escalonamento de tarefas no Grid ou mesmo nas definições relacionadas com o design dos workflows em ambientes de Grid.

2.4 - Ciclo de Vida de Workflows Científicos

O ciclo de vida de um workflow científico, em geral, costuma representar o ciclo de vida do experimento *in-silico* que está sendo reproduzido. Segundo Moreau et al. (2003), o ciclo de desenvolvimento de um workflow científico pode ser dividido em seis etapas.

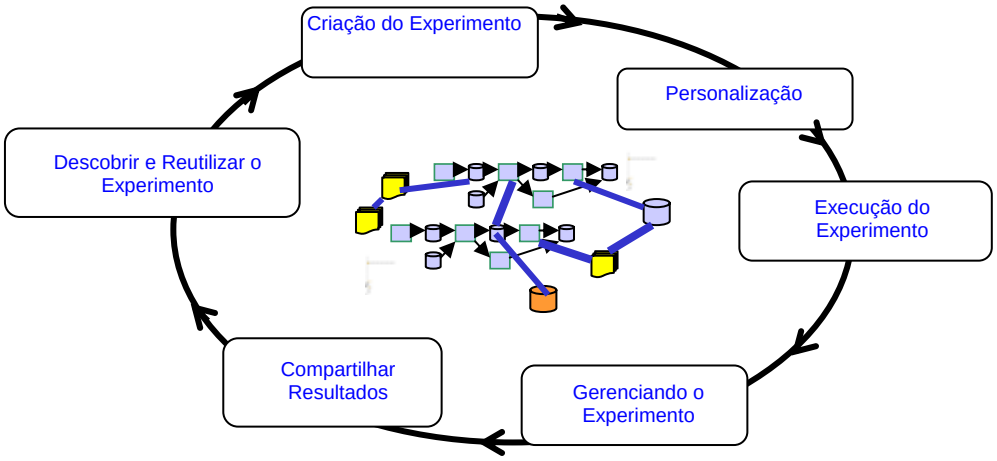

Figura 2.6 - Ciclo de vida de um *workflow* científico, adaptado de ([my]Grid, 2008)

Na figura 2.6 é possível observar o ciclo de vida de um experimento científico. Este ciclo torna possível o processo de descoberta, envolvendo as informações sobre quem executou o experimento, quando, onde e por quê. Além disso, possibilita reutilizar e adaptar trabalhos executados anteriormente pela comunidade científica, além de iniciar o desenvolvimento de um novo experimento. As etapas deste ciclo são:

1. **Criação do Experimento:** etapa na qual é gerado o workflow que representa o experimento que será representado. Visa à validação de uma hipótese através da invocação de serviços.

2. **Personalização:** neste passo devem-se registrar anotações que sejam relevantes para o experimento.

3. **Execução do Experimento:** nesta fase ocorre a execução de cada uma das atividades do workflow, previamente definidas. Esta execução é realizada por uma máquina de workflow. Ocorre também nesta etapa a geração do fluxo de dados.

4. **Gerenciando o Experimento:** aqui ocorrem verificações sobre o andamento do workflow, além de funções como o disparo avisos para cada evento ocorrido ou geração de exceções.

5. **Compartilhar Resultados:** nesta etapa ocorre a publicação do workflow gerado para que possa ser utilizado e avaliado em outros experimentos.

6. **Descobrir e Reutilizar o Experimento**: é um refinamento da fase anterior, onde é possível encontrar um experimento previamente produzido e incorpora-lo a outro experimento.

2.5 - Proveniência de Dados em Workflows científicos

Workflows científicos podem gerar dados na ordem de terabytes aproximadamente, proporcionando a obtenção de metadados que fornecem semântica

aos workflows e possibilitam sua reutilização. Um dos tipos de metadados é a **proveniência dos dados**, também chamada de **linhagem** ou ***pedigree***. Estas informações adicionam um valor significativo para projetos científicos com grande fluxo de dados.

No domínio científico é possível encontrar diferentes formas de proveniência com os mais diversos propósitos. Por exemplo, publicações são comumente utilizadas para representar a proveniência de dados e resultados de um experimento.

Goble (2002) resume as diversas funcionalidades para as informações de proveniência da seguinte maneira:

1. **Qualidade dos Dados:** informações de proveniência podem ser utilizadas para estimar a qualidade e a confiabilidade dos dados baseando-se na origem dos dados e suas transformações (Jagadish et al, 2004).

2. **Auditoria dos Caminhos:** os dados de proveniência podem traçar rotas dos dados (Miles et al, 2005), determinar a utilização de recursos (Greenwood et al, 2003) e detectar erros na geração de dados (Galhardas et al, 2001).

3. **Controle de replicação**: informações de proveniência detalhadas permitem a derivação de dados e ajudam a padronizar a replicação (Foster et al, 2003).

4. **Atribuição:** mantém controle sobre as informações do dono do experimento e seus dados. Também permite a citação (Jagadish et al, 2004) e atribui responsabilidades em caso de dados errados.

5. **Informacional:** permite realizar consultas baseadas nos metadados de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os dados.

2.6 - Gerência de Dados em Workflows Científicos

No contexto científico, a necessidade de ambientes distribuídos é evidente e torna essencial a otimização do fluxo de dados neste tipo de ambiente. Especialmente para workflows científicos, as tecnologias de GRID e Web Services, assim como Grid Services mais recentemente, têm sido exploradas visando facilitar a definição de workflows distribuídos e a orquestração de workflows.

A maioria dos sistemas gerenciadores de workflows, tanto os científicos quanto os comerciais, possuem uma arquitetura com repositórios centralizados. Mas, na prática, esta estratégia se torna ineficiente e introduz limitações para robustez e confiabilidade. Além disso, muitas máquinas de execução de workflows concentraram seus esforços somente no fluxo de controle, deixando de lado o fluxo de dados.

Na medida em que capturar formatos heterogêneos de dados, a natureza distribuída dos workflows científicos e a necessidade de prover um caminho integrado para visualizar dados das diferentes etapas do workflow se tornam cada vez mais necessários, determinadas questões se tornam cada vez mais importantes. São elas, quando se deve

capturar o dado ou se este deve ser armazenado, qual dado deve ser incluído, onde e como armazenar esse dado.

Tipos complexos de dados são comuns na área científica, por isso são necessárias funcionalidades específicas para armazenar e gerenciar os dados científicos. Banerjee (2000), por exemplo, propõe soluções para dados biológicos. Ainda no contexto da bioinformática, Davidson et al. (2001) e GMOD (2008) propõem esquemas genéricos para dados genômicos, abrangendo como e onde armazenar os dados intermediários e tentando integrar os dados resultantes da maioria dos workflows possíveis nesta área, mas não oferecem nenhuma funcionalidade para a gerência de workflows. Também existem ferramentas que oferecem suporte à gerência de workflows, mas não possuem uma gerência completa dos dados intermediários, como o myGrid (Goble et al., 2003) e o Kepler (Ludächer et al, 2006), por exemplo.

Silva e Cavalcanti (2005) identificam três características que devem ser observadas na gerência de dados dos workflows científicos:

1. **Integração de dados**: Os dados gerados durante a execução do workflow requerem um tratamento especial levando em consideração algumas peculiaridades apresentadas no ambiente científico. Cada aplicativo executado por uma atividade de um workflow científico geralmente trabalha com diferentes formatos de arquivos, o que torna ainda maior a dificuldade para a integração dessas ferramentas e, conseqüentemente, compartilhar dados com outros programas envolvidos no workflow também se torna uma tarefa complexa.

2. **Localização dos dados**: Em um ambiente distribuído, os passos do workflow podem estar executando em máquinas diferentes. Isto significa que os dados resultantes podem ser armazenados em qualquer uma dessas máquinas.

3. **Proveniência dos dados**: Como cientistas diferentes podem compartilhar dados e programas neste ambiente distribuído, outra importante questão é prover informações sobre a origem dos dados para que se possa garantir confiabilidade aos experimentos *in-silico*.

As diversas ferramentas que propõem a gerência de workflows científicos, como LabBase (Goodman, 1998), gRNA (Laud et al., 2002), myGrid e Kepler, possuem diferentes mecanismos para gerenciar os dados envolvidos nos workflows.

O LabBase, utiliza um sistema gerenciador de banco de dados para armazenar dados de saída das aplicações envolvidas no workflow, mas não oferece suporte à distribuição de dados. Já as ferramentas gRNA e myGrid, oferecem a integração de fontes de dados distribuídas. Além disso, o myGrid mantém os dados resultantes do workflow, informações de usuários e metadados, mas não considera as questões relacionadas ao gerenciamento de dados durante o processo de definição do workflow.

O Kepler faz uso de um conjunto de serviços para tratamento de dados e acessos aos bancos de dados, mas não oferece suporte à integração dos dados. Além disso,

características para facilitar a captura de dados de proveniência ainda estão sendo desenvolvidas.

2.7 - Especificação e Estruturas dos Workflows

A especificação de um workflow pode ser *abstrata* ou *concreta* (Gorderis et al 2005). Uma definição abstrata (ou modelo abstrato) oferece grande flexibilidade para os cientistas, pois os libera das preocupações relacionadas com os detalhes de implementação, criando uma camada capaz de representar o "comportamento" do workflow. As tarefas de um modelo abstrato são portáveis e podem sem mapeadas em quaisquer serviços (distrubuídos, serviços Web de grid, entre outros), além disso, ela pode ser compartilhada com outros cientistas. Resumindo, ela ser baseada em modelos (templates) que descrevem um dado workflow sem especificar quais recursos serão utilizados na execução.

Em muitos SGWf, a representação abstrata de um workflow é definida através de elementos gráficos, como por exemplo, os diagramas de bloco no Kepler e ^{my}Grid. Essa metáfora visual aumenta a percepção dos nuances relacionadas à semântica do modelo. Por outro lado, um modelo concreto, está intimamente à uma dada tecnologia, ele associa quais recursos computacionais são requeridos para a excução das tarefas do workflow. Os conceitos de workflows abstrato e concreto são um ponto de discórdia entre os diversos autores. Por exemplo, os trabalhos de SGWf em grid apresentam diferentes definições para os workflows abstratos. (Ludascher, Altintas, Gupta,2002), (Goderis et al 2005).

As estruturas do Workflow (Aalst, Hee, 2002) indicam o relacionamento temporal entre as componentes do workflow. Um workflow pode ser representado como sendo do tipo *DAG* (*Directed Acyclic Graph*) e *não-DAG*. Workflows DAG contém estruturas (controle, comunicação e dados) do tipo seqüenciais, paralelas ou livres. Esse tipo de estrutura é representado como um grafo, onde cada tarefa é um nó e cada nó pode ter um número arbitrário de filhos. As seqüencias dos nós definem a ordem de execução das tarefas. As estruturas paralelas indicam que as tarefas podem ser executadas concorrentemente. Os workflows não-DAG incluem estruturas de iteração (também conhecido como loops ou ciclos). Elas permitem que os workflow executem tarefas ou sub-workflows de forma repetida. As estruturas serão apresentandas em detalhes na seção 4 onde discutiremos em detalhes os métodos formais de representação de workflows.

3 - Workflows Científicos X Comerciais

De acordo com o que foi apresentado nos itens anteriores, é possível afirmar que os SGWf comercial e científico compartilham algumas características comuns, mas por outro lado eles também apresentam suas especificidades. As principais diferenças entre os SGWfs comerciais e científicos[5] estão representadas na tabela 3.1, elas foram compiladas a partir de uma série de trabalhos, a saber: (Weske, Vossen, Medeiros, 1996), (Singh, Vouk,1998), (Aalst, Hee, 2002), (Sandeep, 2002), (Slominski, Gannon, Fox, 2003), Altintas et al. (2006), Aalst, Dumas, Hofstede (2004), (Rana, 2005), (Sangeeta, 2005), (Yu, Buyya, 2005).

Caraterísticas	SGWf Científico	SGWf Comercial
Quem desenha os Wfs	Cientistas e pesquisadores	Especialistas no negócio ou informatas
Natureza dos fluxos	Dados (maior parte), Controle (menor parte)	Dados (menor parte), Controle (maior parte)
Tipos dos fluxos	Lineares (pipelines) e não lineares (workflows)	Não lineares
Tipos de dados	Arquivos heterogêneos e altamente complexos	Documentos bem estruturados, em geral pequenos
Volume de dados manipulados	Alto	Baixo
Freqüência de mudanças nos Wfs	Elevada	Baixa
Volume de Processamento	Elevado	Médio (pode variar em função da natureza do negócio)
Permitem a intervenção humana	Sim, alguns Wfs podem ser executados durante longos períodos de tempo, logo, eles podem ser constantemente avaliados. As condições de intervenção devem ser definidas durante a criação do Wf	Sim, raramente o processamento é menos intenso e demorado
Validação de dados intermediários	Sim, checagem de consistência e validações é importante para a continuação do processamento	Desconhecido
Interoperabilidade com outros SGWfs	Baixo, Desejável, porém existem muitas iniciativas concorrentes	Desconhecido
Validação dos dados/resultados	Desejável, porém atualmente é baixa	Desconhecido
Tolerâncias à falhas	Desejável, porém atualmente é baixa	Desejável, as falhas podem afetar os processos de negócio
Executa em ambientes distribuídos	Sim, apenas alguns SGWf suportam essa necessidade	Sim, principalmente quando se trata de serviços Web
Wf executado de maneira rotineira	Sim, porém apresenta interdependência com o projeto de pesquisa	Sim
Aderência a padrões de mercado	Baixa	Moderada
Execução parcial	Desejável e necessário, porém nem sempre suportado	Desnecessário
Modificação dinâmica	Desejável, em função dos resultados obtidos, alterações podem ser necessárias	Indesejável, são estáticos por definição

[5] Nesta tabela consideramos que os SGWfs em grade são casos especiais de SGWfs científicos, por este motivo não fazemos nenhuma distinção entre eles.

Suporte ao reuso	Desejável, parte de um Wf pode ser aproveitado por outros pesquisadores no mesmo experimento ou em experimentos futuros	Não é crucial, os Wfs são criados para tarefas específicas
Suporte a proveniência e persistência de dados	Sim, É desejável que se armazenem informações as fontes de dados utilizadas, sobre as execuções dos Wfs e seus resultados produzidos	Desconhecido
Geração de logs de utilização	Sim, mantém informações sobre como o pesquisador conduziu um determinado experimento, essas informações podem ser utilizadas tanto no reuso de Wfs quanto na validação do experimento	Sim, é uma característica desejável para tratar questões de auditoria, segurança e depuração de erros
Suporte à anotação	Desejável, porém nem sempre suportado, cada vez mais essa característica é apreciada nos SGWf	Desconhecido
Capaz de utilizar serviços Web	Necessário, porém nem sempre suportado	Desejável, porém nem sempre suportado
Suporte ao repositório central de serviços	Desejável, entretanto ele deverá ser transparente para o usuário, ocultando detalhes de implementação. Pode ser um repositório físico ou virtual	Desejável, porém nem sempre suportado
Suporte a subtarefas	Desejável, facilita o reuso, porém nem sempre suportado	Desconhecido
Suporte à Web semântica	Desejável, porém nem sempre implementado	Desconhecido
Suporte a pré e pós - processamento de dados	Sim, desejável, porém nem sempre suportado	Sim
Suporte a transformação de dados	Sim, fundamental, os dados científicos são muito heterogêneos	Desconhecido
Suporte a ambientes distribuídos (Grids, P2P, Cluster)	Desejável, porém nem sempre implementado	Desconhecido
Apresenta um ciclo de vida	Sim, porém é reduzido, deve-se à natureza do método científico	Sim, maior e estável
Aplica-se a um domínio específico	Em alguns casos sim, porém existem muitos exemplo de workflows interdisciplinares	Sim, sempre
Aprendizado com os erros	Sim, o processo de definição de workflows é baseado na tentativa e erro fazendo com que a análise e armazenamento de resultados negativos sejam tão importantes quanto os resultados positivos.	Desconhecido
Documentação	Sim, é desejável porém nem sempre é produzidas pelos cientistas ou pelos SGWf	Desconhecido, porém estima-se que as empresas mantenham algum tipo de documentação que pode ser utilizada nas manutenções

Tabela 3.1 – Comparativo entre SGWfs científicos e comerciais

O termo desconhecido presente na tabela 3.1 deve-se ao fato de nem sempre encontrarmos um equivalente nos workflows comerciais. Finalmente é importante ressaltar dois pontos importantes: (i) A tabela foi estabelecida após a leitura de inúmeras referências, uma vez que não há uma separação clara entre os requisitos de cada tipo de SGWf; (ii) os critérios podem variam de autor para autor.

4 - Ferramental teórico de apoio (formalismos)

A modelagem de Workflows consiste na criação de especificações para os mesmos, tais especificações são usadas como entrada para a execução das tarefas ali descritas, por intermédio de um sistema de gerência de workflows. O uso de especificações formais tem sido visto com bastante interesse pela comunidade acadêmica e de tecnologia, visto que métodos formais reduzem a ambigüidade e abrem a possibilidade para verificação e análise.

Diversos trabalhos vêm sendo propostos no intuito de prover formalismo para descrições de workflows. Esses trabalhos fazem uso de diferentes técnicas, tais como: *máquinas de estados finitos*, *álgebra de processos*, *máquinas de estados abstratos*, *lógica temporal*, *lógica transacional* e *redes de Petri*. Tais formalismos vêm sendo estudados desde a década de 60, e oferecem diferentes formas para fazer verificação e análise de workflows.

Neste trabalho estaremos voltados para as técnicas de álgebra de processos e redes de Petri, as quais serão discutidas nas subseções seguintes.

4.1 - Álgebra de Processos

Para entendermos o significado do termo *álgebra de processos*, primeiramente, precisamos definir o que seria um *processo*. Um processo está relacionado ao comportamento de um sistema, em particular a execução de um software, as ações de uma máquina ou até mesmo as ações de um ser humano. O termo álgebra indica uma visão algébrica/axiomática do comportamento desse sistema (Baeten, 2004).

O comportamento de um sistema pode ser descrito como o número de eventos ou ações que um sistema pode executar, envolvendo a ordem de execução, além de outros aspectos, tais como probabilidade e tempo.

Considerando as propriedades de uma álgebra de processos, podemos vê-la como uma ferramenta poderosa que permite a descrição de um sistema, abrangendo a interação de seus eventos.

A história das álgebras de processos tem início na década de setenta, pois, até então, a teoria de redes de Petri era a única parte da teoria da concorrência que existia, tendo sido criada a partir de 1962.

A partir da década de setenta, três estilos de raciocínio formal a cerca de programas de computador, consolidam-se como forma de oferecer semântica a linguagens de programação. O primeiro estilo é denominado *semântica operacional*, na qual um programa de computador é modelado como a execução de uma máquina abstrata, onde um estado da máquina corresponde a uma avaliação de variáveis, e uma transição entre estados corresponde a uma instrução de programa. Os primeiros trabalhos desse estilo foram de autoria de McCarthy (McCarthy, 1963). A *semântica denotacional* é o segundo estilo, sendo mais abstrata que a semântica operacional, os programas de computador são modelados por uma função que transforma entrada em saída. Os trabalhos mais relevantes desse estilo pertencem a Scott e Strachey (1971). A *semântica*

axiomática é o terceiro estilo, e dar ênfase à geração de métodos de prova para a corretude de programas. Os pioneiros nesse estudo são Floyd e Hoare (1969).

No início dos anos setenta Hans Bekič fez parte da equipe que trabalhava na definição e semântica (denotacaional) de ALGOL e PL/I. A partir do trabalho com linguagens de programação, Bekič passou a tratar do problema em como dar semântica denotacional para composição paralela.

Em 1973, Robin Milner desenvolveu a teoria de processos denominada CSS (Calculus of Communicating Systems). Milner continuou a desenvolver suas teorias gerando novos trabalhos e servindo de base para outros como a CSP de Tony Hoare, que se baseia nos princípios de Milner, que tem sua base fixada nas teorias de concorrência, paralelismo e distribuição de sistemas computáveis.

Desde seu surgimento até os dias atuais, diversas extensões vêm sendo propostas, incluindo a abordagem de tempo, probabilidade e prioridade. De uma forma geral, as álgebras de processos podem ser vistas como linguagens algébricas que permitem a descrição de sistemas distribuídos e concorrentes e a verificação formal de suas propriedades (Costa et al., 2005). Dessa maneira, sua aplicabilidade abrange diversas áreas, podendo ser usada, por exemplo, para modelar experimentos científicos descritos como workflows.

Nessa direção, alguns trabalhos estudam a aplicabilidade de álgebra de processos como base formal para a descrição de workflows. A seção a seguir apresenta o mapeamento de padrões de WF para álgebra de processos encontrado no trabalho de (Braghetto, 2005).

4.1.1 - Álgebras de Processo aplicadas a WFs

Em (Braghetto, 2005) é descrito o mapeamento dos padrões de workflows propostos por van der Aalst para a linguagem de definição de planos (NPDL) usando Álgebra de Processos como base formal. Nesta seção, daremos ênfase ao mapeamento entre os padrões de workflows e álgebra de processos. Os padrões são apresentados com uma breve descrição, seguidos de uma representação gráfica em redes de Petri e seu respectivo mapeamento em álgebra de processos.

4.1.1.1 - Padrões de Fluxo de Controle

Os padrões de fluxo de controle incluem workflows com padrões sequeciais, paralelos e de sincronização.

Padrão 1 – Seqüência

A atividade B é habilitada após o término da atividade A.

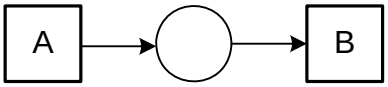

Figura 4.1 – Seqüência das atividades A e B.

Álgebra de Processos: **P = A.B**

Padrão 2 – Divisão Paralela

Após o término da atividade A, ambos B e C podem ser executadas em paralela.

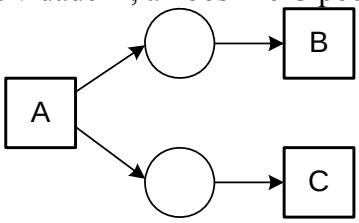

Figura 4.2 – Divisão paralela entre as atividades B e C.

Álgebra de Processos: **P = A . (B || C)**

Padrão 3 – Sincronização

A atividade C é habilitada apenas após o término de ambas as atividades A e B.

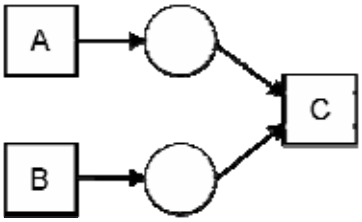

Figura 4.3 – Sincronização para a atividade C.

Álgebra de Processos: **P = (A || B) . C**

Padrão 4 – OU Exclusivo

Após o término da atividade A, apenas uma das atividades B e C serão implicitamente escolhidas para ter sua execução habilitada.

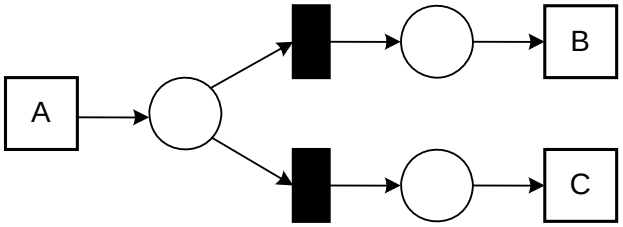

Figura 4.4 – OU exclusivo entre as atividades B e C.

Álgebra de Processos: **P = A . (B + C)**

Padrão 5 – Merge Simples

No início, ambas as atividades A e B estão habilitadas; após o término de uma delas, C é habilitado.

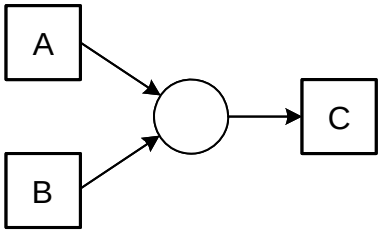

Figura 4.5 – Merge simples.

Álgebra de Processos: **P = (A + B) . C**

4.1.1.2 - Padrões de Sincronização avançada

Padrão 6 – Multi-Choice

No início, três diferentes opções serão habilitadas: a divisão paralela de B e C, apenas a execução de B e apenas a execução de C.

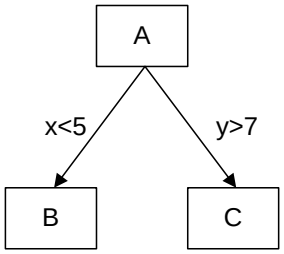

Figura 4.6 – Multi-choice entre B, C, B e C (paralelo).

Álgebra de Processos: **A . (B || C + B + C)**

Padrão 7 – Synchronizing Merge

Se no padrão 6 a opção escolhida foi a divisão paralela de B e C, então o synchronizing merge habilita a atividade D após o término das atividades B e C. Entretanto, se a opção escolhida foi B, então a atividade D é habilitada após o término de B. Similarmente, se a opção escolhida foi C, então a atividade D é habilitada após o término de C.

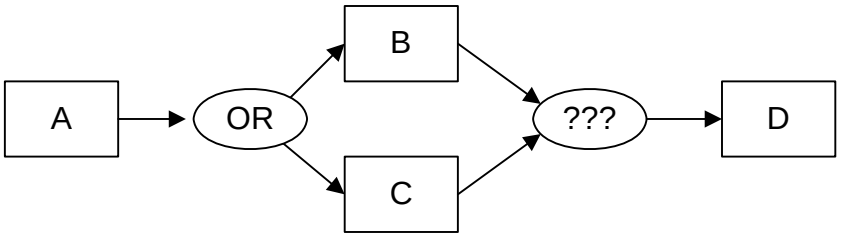

Figura 4.7 – Merge sincronizado.

Álgebra de Processos: **P = A.(B || C + B + C).D**

Padrão 8 – Multi-Merge

Se a opção escolhida foi a divisão paralela de B e C, então o Multi-Merge executa a atividade D uma vez para cada término de uma thread de controle na divisão paralela. Por outro lado, se a opção escolhida foi a execução de B, então a atividade D é habilitada após o término de B. Similarmente, se a opção escolhida foi a execução de C, então a atividade D é habilitada para o término de C.

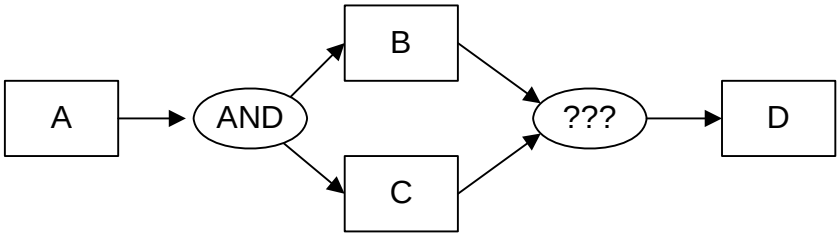

Figura 4.8 – Multi-Merge.

Álgebra de Processos: **A.((B.D) || (C.D) + B.D + C.D)**

Padrão 9 – Discriminator

Se no multi-choice a opção escolhida foi a divisão paralela de B e C, então o multi-merge habilita a atividade D após o término da thread de controle na divisão paralela que terminar primeiro. Por outro lado, se a opção escolhida foi a execução de B, então a atividade D é habilitada após o término de B. Similarmente, se a opção C foi escolhida, então a atividade D é habilitada após o término de C.

Álgebra de Processos: **A.((B.D) || C + B||(C.D) + (B||C).D + B.D + C.D)**

Padrão 10 – Ciclos Arbitrários

Um ponto no processo do workflow onde uma ou mais atividades podem ser feitas repetidamente.

Álgebra de Processos: Obtêm-se os ciclos por meio do uso de equações recursivas.

Padrão 11 – Terminação Implícita

Um dado subprocesso deve ser finalizado quando não houver mais nada para ser feito.

Álgebra de Processos: Suporta este padrão diretamente. Não existe um termo explícito que represente o estado final de um processo.

4.2 - Redes de Petri

Redes de Petri constituem um modelo de representação formal para sistemas dinâmicos. As redes originais foram desenvolvidas por Petri (1962), posteriormente, várias extensões foram propostas. Algumas dessas extensões provêem uma modelagem mais fácil, mantendo o mesmo poder de expressividade, enquanto que algumas outras possuem maior poder de expressividade.

A teoria de redes de Petri apresenta as seguintes propriedades: *formalização matemática*, *representação gráfica* e *analisabilidade*. Assim, as redes permitem a representação com facilidade de todas as relações de causalidade entre processos em situações de seqüencialidade, conflito, concorrência e sincronização. A aplicabilidade de redes de Petri estende-se por um grande número de áreas, incluindo protocolos de comunicação, avaliação de desempenho e sistemas distribuídos.

O modelo clássico de uma rede de Petri é formado por dois tipos decomponentes: um ativo denominado *transição* (representado graficamente por um retângulo ou barra) e outro passivo denominado *posição* (representado graficamente por um círculo) e *arcos direcionados*. As posições correspondem às variáveis de estado e as transições às ações (eventos) realizadas pelo sistema. Os arcos interligam posições às transições, e correspondem à relação entre as condições verdadeiras, que num dado momento, possibilitam a execução das ações, enquanto os arcos que interligam transições às posições representam a relação entre as ações e as condições que se tornam verdadeiras com a execução das ações.

A definição formal de uma rede de Petri consiste numa tupla $PN = (P, T, F, M_0)$, onde P é um conjunto de posições, T é um conjunto de transições, $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos (relações de fluxo) das posições às transições e vice-versa. $M_0 : P \rightarrow N^+$ é o marco inicial no qual para cada posição $p \in P$ existem $n \in N$ tokens. Os conjuntos de posições e transições são disjuntos e uma rede de Petri deve conter ao menos uma posição e uma transição: $P \cap T = \varnothing$, $P \cup T \neq \varnothing$.

Uma rede de Petri é um modelo abstrato e pode ser utilizado para representar diferentes situações. Todavia, variantes de redes de Petri vêm sendo desenvolvidas ao longo do tempo, no intuito de facilitar a modelagem de determinadas situações ou estender a expressividade das redes clássicas. A seguir, algumas dessas extensões são apresentadas de forma breve.

Redes de Petri com pesos permitem modelagem de redes de Petri clássicas numa maneira mais sucinta, onde pesos são adicionados aos arcos para indicar que múltiplos tokens são consumidos e produzidos durante o disparo de uma transição. Um arco com peso *k* corresponde a *k* arcos paralelos em redes clássicas.

Redes de Petri coloridas permitem modelar a identidade de tokens individualmente anexando valores (ou cores) aos tokens. Tal propriedade permite uma modelagem mais detalhada dos objetos que os tokens representam na rede de Petri.

Redes de Petri temporizadas permitem a modelagem do comportamento temporal de um sistema. Existem diversas maneiras de introduzir tempo em redes clássicas de Petri. A estratégia mais comum consiste em associar um timestamp a cada *token* (indicando o momento em que o *token* está disponível para *consumo*) e associar um tempo a cada transição (indicando a mudança para o timestamp de tokens produzidos). Este tipo de modelagem permite analisar indicadores de desempenho quantitativos tais como tempo de resposta, tempo de ocupação, e tempo de *throughput*.

Redes de Petri hierárquicas permitem modelagem composicional, onde uma transição pode não ser apenas atômica, mas também complexa, nesse caso há alguma subrede que define esta tarefa complexa. Neste caso, uma rede de Petri pode conter outra rede de Petri como token, introduzindo o conceito de redes de Petri aninhadas que se comunicam ao sincronizar transições entre diferentes níveis .

4.2.1 - Redes de Petri aplicadas a Workflows

Há diversas razões para usar redes de Petri em modelagem de workflow, em (Aalst, 1998) destaca as seguintes:

Semântica Formal

Um processo de workflow especificado em termos de uma rede de Petri possui uma definição clara e precisa, porque a semântica de redes de Petri e suas extensões (colorida, temporizada, hierárquica) foram definidas formalmente.

Natureza Gráfica

Redes de Petri são uma linguagem gráfica. Como um resultado, redes de Petri são intuitivas e fáceis de aprender. A natureza gráfica também suporta a comunicação com usuários finais.

Expressividade

Redes de Petri suportam todas as primitivas necessárias para modelar um processo de workflow. Todas as construções de roteamento presentes em sistemas de gerência de workflow de hoje podem ser modeladas. Além disso, o fato dos estados serem representados explicitamente, permite a modelagem de escolhas implícitas.

Propriedades

Nas últimas três décadas muitas pessoas têm investigado as propriedades básicas de redes de Petri. Como resultado, há uma grande quantidade de conhecimento, em forma de livros e artigos, que exploram esta modelagem.

Análise

Redes de Petri são marcadas pela disponibilidade de muitas técnicas de análise. Claramente, isto é uma grande vantagem em favor do uso de redes de Petri para modelagem de workflow. Estas técnicas podem ser usadas para prover propriedades e calcular as medidas de desempenho (tempo de resposta, tempo de espera, taxas de ocupação, etc.).

Independente de Fornecedor

Redes de Petri provêem um framework independente de ferramenta para modelagem e análise de processos. Redes de Petri não são baseadas num pacote de software de um fornecedor específico e não correm o risco de deixarem de serem produzidos.

Uma classe especial de redes de Petri de alto nível é idenificada por van der Aalst (1998) como apropriada para a modelagem de workflows. Essas redes de workflow têm duas propriedades especiais: possuem exatamente uma posição de entrada e uma posição de saída, e cada transição e cada posição estão num caminho da posição de entrada da rede até a posição de saída da rede.

De acordo com van der Aalst (2002) e Russel et al (2006), um workflow pode ser visto em três dimensões: a dimensão do caso, a dimensão do processo, e a dimensão do recurso. A dimensão de caso indica que um workflow consiste de diferentes casos que são tratados individualmente, isto é, cada pedaço do processo é executado por caso

específico (exemplos de casos são: um financiamento, uma declaração de imposto, um pedido). A dimensão de processo indica que um workflow consiste de diversas tarefas em alguma ordem de roteamento, e a dimensão de recurso indica que as tarefas num workflow são executadas por alguns recursos organizacionais. Um recurso pode ser uma máquina (impressora, fax, etc.) ou uma pessoa (participante, empregado, etc.).

Tal workflow pode ser modelado numa rede de workflow, da seguinte maneira: os casos são modelados por tokens de alto nível, que tem uma identidade, são distinguíveis uns dos outros, e podem conter estrutura de dados; as tarefas são modeladas por transições e as condições são modeladas por posições. Um modelo de workflow pode ser construído especificando como os casos são roteados juntamente com as tarefas que precisam ser executadas. De acordo com o WfMC (WfMC, 2008), são definidas as seguintes construções de roteamento: seqüência, paralela, condicional e iteração. A seguir, é apresentada uma breve discussão a cerca de algumas dessas construções de roteamento e seu mapeamento para redes de Petri.

Roteamento Seqüencial

É usado para lidar com relacionamentos causais entre tarefas. Considerando duas tarefas A e B. Se a tarefa B é executada depois do término da execução da tarefa A, então A e B são executadas seqüencialmente. A figura 4.9 mostra que o roteamento seqüencial pode ser modelado através da adição de posições. A posição c2 modela o relacionamento causal entre a tarefa A e a tarefa B, isto é, c2 representa uma pós-condição para a tarefa A e uma pré-condição para a tarefa B.

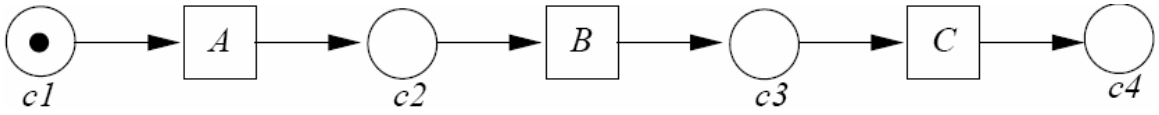

Figura 4.9 – Roteamento Seqüencial

Roteamento Paralelo

É usado em situações onde a ordem de execução é menos restrita. Por exemplo, duas tarefas B e C precisam ser executadas, mas a ordem de execução é arbitrária. Para a modelagem em redes de Petri são construídos dois blocos: o AND-split e o AND-join.

A figura 4.10 exibe a rede do roteamento paralelo, onde a execução do AND-split habilita ambas as tarefas B e C, enquanto que AND-join indica que D só estará habilitado após a execução de ambos B e C, isto é, D é usado para sincronizar dois subfluxos. Como resultado, as tarefas B e C são executadas em paralelo.

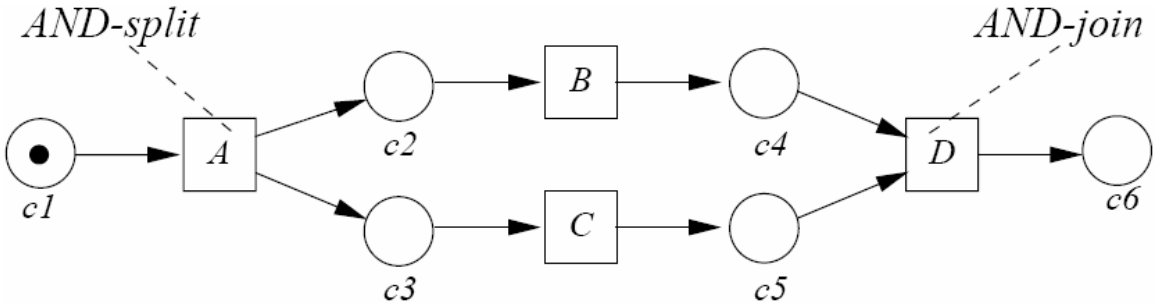

Figura 4.10 – Roteamento Paralelo

Roteamento Condicional

É usado para permitir que um roteamento possa variar entre casos. Para modelar uma escolha entre duas ou mais alternativas, dois blocos de construção são usados: o OR-split e o OR-join. A figura 4.11 ilustra a situação onde a tarefa A é seguida ou pela tarefa B ou pela tarefa C, ou seja, é feita uma escolha entre B e C. A execução de uma dessas duas tarefas é seguida pela execução da tarefa D.

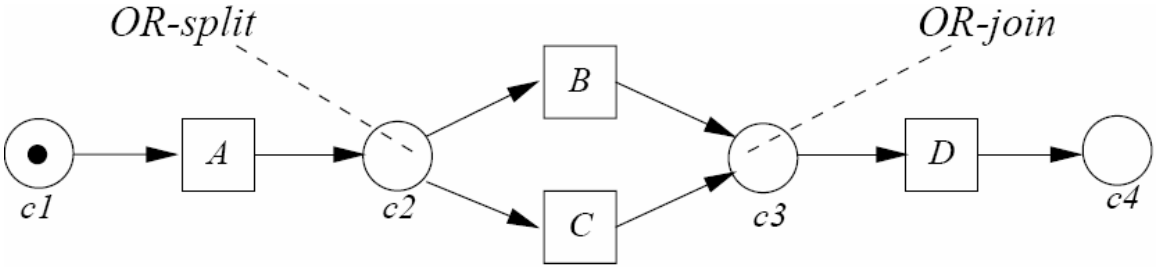

Figura 4.11 – Roteamento Condicional

Problemas no uso de redes de Petri em WF

De acordo com Eshuis e Dehnert (2003) os sistemas de gerência de workflows são reativos, e tal fato implica que a semântica de redes de Petri não é completamente capaz de modelar esses sistemas. A semântica *token-game* das redes de Petri não conseguem modelar o comportamento de sistemas reativos, e portanto, não são capazes de modelar o comportamento de uma máquina de execução de worfkflows.

Um sistema de gerência de workflows não executa atividades, mas meramente coordena a execução dessas atividades que são realizadas por atores humanos ou sistemas de software. Sendo assim, um sistema de gerência de workflows não controla o ambiente, mas reage a eventos no ambiente (início de casos, término de tarefas agendadas) através da criação de certos efeitos no ambiente (inicialização de instâncias de atividade, agendamento de tarefas). Um sistema reativo é normalmente representado através do uso de regra ECA (Evento-Condição-Ação), especificando as ações a que o sistema deve responder na ocorrência de eventos.

Em Eshuis (2002) descreve diversos problemas no uso de redes de Petri para modelagem de sistemas de gerências de workflows. O principal problema levantado no trabalho é que a distinção entre o ambiente e o sistema de gerência de workflow não é capturada.

De acordo com van der Aalst e ter Hofsted (2005), redes de Petri possuem três limitações quando usadas para a modelagem de workflows. Essas limitações se fazem presentes nos padrões que envolvem *instâncias múltiplas*, *sincronização avançada* e *cancelamento*.

Padrões envolvendo múltiplas instâncias ocorrem em workflows onde algum caso filho é instanciado um número de vezes. O número de instanciações desses casos filhos podem ser definidos durante tempo de projeto ou durante o tempo de execução. Por exemplo, durante um processo de revisão de uma conferência, para cada artigo submetido o número de revisores é selecionado e pede-se suas opiniões; estes revisores são casos filhos do paper submetido, e é necessário sincronizar o resultado destes casos filhos. Estes padrões são problemáticos visto que a execução do workflow precisa manter um log e sincronizar essas múltiplas instâncias, para manter o workflow consistente. É possível modelar tais padrões usando redes de Petri, mas é bastante complicado. Torna-se preciso sincronizar todos os casos filhos de um caso, mas sem confundi-los com casos filhos de outro caso. Além disso, é preciso manter um log do número de instâncias ativas de casos filhos.

Os padrões de cancelamento ocorrem no workflow onde alguma atividade deve levar ao cancelamento de casos do workflow. Cancelamentos podem ocorrer em momentos variados e devem ter o efeito desejado independente do estado da execução do workflow. O problema é que em redes de Petri todas as regras são locais para uma única transição ou estado, enquanto que o cancelamento produz efeitos globais. Ou seja, não importa em que estado o workflow se encontre, o caso inteiro deve ser cancelado.

5 - Sistemas de Gerência de Workflow

A comunidade científica demonstra grande interesse pelos workflows científicos, esse fato pode ser facilmente comprovado pelo grande número interesse órgãos de fomento, além de conferências e simpósios. Os workflows científicos desempenham um importante papel em um grande número de projetos, dentre os quais se destacam os projetos PtolemyII-Kepler, myGrid, além de uma série de tecnologias de apoio.

5.1 - Tecnologias de Apoio

BPEL já é o padrão sstabelecido como a linguagem para a especificação de processos de negócio, além disso, o BPEL vem ganhando espaço na especificação de processos científicos (Emmerich, 2006).

5.1.1 - A Linguagem BPEL

BPEL, Bussines Process Language, é uma linguagem para a especificação de processos de negócios que descreve processos assíncronos de longa duração e suas interações. De forma simplificada, o BPEL pode ser visto como uma linguagem de programação que é expressa em XML e que provê as construções de controle de fluxo para a combinação de serviços WSDL. A linguagem é focada em Serviços Web, entretanto outros elementos podem ser incorporados a um fluxo de processos BPEL (ex. métodos Java, EJBs etc.)

Um processo descrito em BPEL representa um conjunto de comportamentos públicos observáveis, inclui informações de quando esperar uma mensagem, quando enviar uma mensagem e quando tratar operações que falharam.

A especificação BPEL baseia-se em três conceitos(Curbera, 2003):

- Protocolos de descrição de processos invariavelmente expressam comportamentos dependentes de dados. Neste caso a linguagem necessita de construtores condicionais e de expiração de tempo (*time-out*);
- A habilidade de especificar condições de exceção e suas conseqüências, tais como seqüências de recuperação, são tão importantes para a linguagem de processos como as seqüências que tratam fluxos corretos de execução;
- Interações de longa duração incluem com freqüência unidades de trabalhos múltiplas, aninhadas e cada uma com seus próprios requisitos de dados.

5.1.2 - Estrutura da Linguagem BPEL

Um processo BPEL é normalmente divido em duas seções: uma seção de declaração e uma seção contendo os processos de "atividades". Isto envolto em um elemento XML <process> que identifica o nome do processo em si.

As declarações globais são tipicamente a primeira seção de um processo que contém as declarações globais do workflow, isto inclui os serviços Web que serão utilizados, os

parceiros do workflow (<partnerLinks>), quais variáveis serão utilizadas, associadas a tag <variable>, além das definições dos tipos complexos que serão utilizados, tag <types>.

Abaixo apresentamos um extrato das declarações globais de um processo que recupera registros do Entrez informado um ID primário (Entrez, 2008).

```
<process name = "LoanFlow"
        targetNamespace = http://samples.cxdn.com
         ...>
<partnerLinks>
    <partnerLink name = "EntrezFetchService"
                 partnerLinkType = "services:eUtilsServiceSoap"
                 myRole = "EntrezSpellRequester"
                 partnerRole = "EntrezSpellProvider"/>
    <partnerLink name = "EntrezSpellService"
                 partnerLinkType = "services:eUtilsServiceSoap"
                 myRole = "EntrezFetchRequester"
                 partnerRole = "EntrezFetchProvider"/>
</partnerLinks>

<variables>
    <variable name = "term"
              messageType = "services:spellRequestMessage"/>
    <variable name = "outputSpell"
              messageType = "services:spellResultMessage"/>
    <variable name = "outputFetch"
              messageType = "services:fetchResultMessage"/>
    <variable name = "queryKey"
              messageType = "services:queryKeyRequestMessage"/>
</variables>
<!—Process definition follows -->
...
</process>
```

As variáveis inputClient e outpuClient são necessárias para a entrada e saída do workflow.

Outros construtores de alto nível também são declarados nesta primeira seção. São eles os construtores de erro global, expressos na tag <faultHandlers>, e os manipuladores de falha de transação global, expresso por <compensationHandlers>.

A segunda parte do processo BPEL que contém a lógica de processamento e a área de definição do processo. Ela contém os passos com chamadas a serviços Web que são interconectados para compor um processo útil ao usuário. Nesta seção existem dois tipos de atividades.

O primeiro tipo é referente às atividades de processo e atividades de dados. As atividades de processo atuam na chamada e recepção dos serviços Web, como exemplos destas primitivas têm-se as tags: <invoke>, <receive> e <reply>. Ainda nas atividades de processo podemos controlar o mesmo, através das tags : <wait> e <terminate>. Para a manipulação de dados temos a tag <assign>.

As atividades estruturadas definem o controle programático sobre quais passos serão executados no workflow, incluem tags para o tratamento de condições com a tag

<case>; execução de laços, <while>; construtores de execução em paralelo como o <flow>; e construtores de execução seqüencial com o <sequence>.

```
<flow>
      <sequence>
        <invoke name="invokeEntrezFetch"
                partnerLink="EntrezFetchService"
                portType="services:eUtilsServiceSoap"
                operation="run_eFetch"
                inputVariable="queryKey"/>
        <receive name="receive_invokeEntrezFetch"
                partnerLink="EntrezFetchService"
                portType="services:eUtilsServiceSoapCallback"
                operation="run_eFetch_MS"
                variable="outputFetch"/>
      </sequence>
      <sequence>
      <invoke name="invokeEntrezSpell" partnerLink="EntrezSpellService"
                portType="services:eUtilsServiceSoap"
                operation="run_eSpell"
                inputVariable="term"/>
        <receive name="receive_invokeEntrezSpell"
                partnerLink="EntrezSpellService"
                portType="services:eUtilsServiceSoapCallback"
                operation="run_eSpell_MS"
                variable="outputSpell"/>
      </sequence>
    </flow>
```

O código BPEL acima ilustra a invocação do serviço entrezSpell e entrezFetch. O primeiro retorna sugestões de escrita para um dado termo biológico; o segundo retorna uma lista de registros dado um identificador primário. Para a execução de cada serviço temos as tags <invoke> e <receive> dentro de uma seqüência (<sequence>), garantindo que eles serão executados um após o outro. Mas as duas seqüências de atividades estão envoltas por uma atividade de fluxo (<flow>), o que permite que as subseqüências sejam executadas em paralelo.

5.1.3 - Ferramentas para a construção e execução de Workflows BPEL

Engines de Execução

Os engines de execução de Workflow BPEL são na realidade a versão atualizada, para serviços Web, dos sistemas de gerenciamento de Workflow. Estes sistemas aderem as especificações do SGWf, e possuem arquitetura baseada nos padrões definidos pelo SGWf, assim na análise dos engines BPEL podemos ter como referência a arquitetura proposta e apresentada na seção anterior.

Os engines que se destacam comercialmente e academicamente para a execução de workflows em BPEL são ActiveBPEL, JBoss jBPM, Microsoft BizTalk Server, IBM Websphere Process Server e o Oracle BPEL Process Manager. Uma lista mais completa de engines BPEL pode ser encontrada na wikipedia (Wikipedia, 2008). No conjunto acima podemos separar os engines em dois grandes grupos: comerciais e gratuitos. Entre os comercias temos o BizTalk, IBM WebSphere Process Server e o Oracle BPEL PM. Os gratuitos são o ActiveBPEL e o JBoss jBPM; é importante observar que o JBoss jBPM, assim como o ActiveBPEL, é gratuito para o uso, sobre a Licença LGPL

que permite acesso ao fonte do engine, mas não restringindo o uso deste dentro de aplicações comerciais.

Nas próximas subseções apresentamos os detalhes de cada destes engines.

Oracle BPEL Process Manager

O Oracle BPEL Process Manager(Oracle, 2008) encontra-se na sua versão 2.0 tem como foco principal a integração de serviços de forma colaborativa e transacional para processos de negócio. Ele incorpora as seguintes características:

- Suporte nativo ao BPEL;
- Integra serviços Web, mensagens JMS, componentes JCA e tarefas do usuário;
- Monitoração e auditoria;
- Compatibilidade com a especificação BPEL4WS 1.1;
- Permite a troca de mensagens síncronas e assíncronas;
- Instalação em menos de 15 minutos.

O núcleo do engine BPEL da Oracle agrega outras características que tornam a ferramenta bastante robusta. A primeira característica é a "desidratação do contexto" de execução do workflow que é a persistência dos estados da execução do workflow. Esta característica torna a execução do workflow mais robusta, pois caso haja falha na execução de algum processo, o workflow pode ser reiniciado a partir do ponto onde houve a falha. Outra característica não menos importante é a capacidade de manipular arquivos XML grandes. Esta característica introduz um componente de gerenciamento de banco de dados ao engine, expandindo as capacidades da ferramenta. Por fim, a infra-estrutura do Oracle BPEL ainda conta com um servidor UDDI para facilitar a localização de serviços Web conhecidos.

Como infra-estrutura de suporte ao engine Oracle BPEL, as possibilidades são os servidores de aplicação J2EE mais tradicionais, tal como, o Oracle AS, o Weblogic da BEA, IBM Websphere Application Server e o JBoss. A possibilidade de instalação do engine em diversos servidores de aplicação aumenta o espectro de instalações existentes em que a ferramenta pode ser instalada. A seguir apresentamos os principais componentes do Oracle BPEL Process Manager.

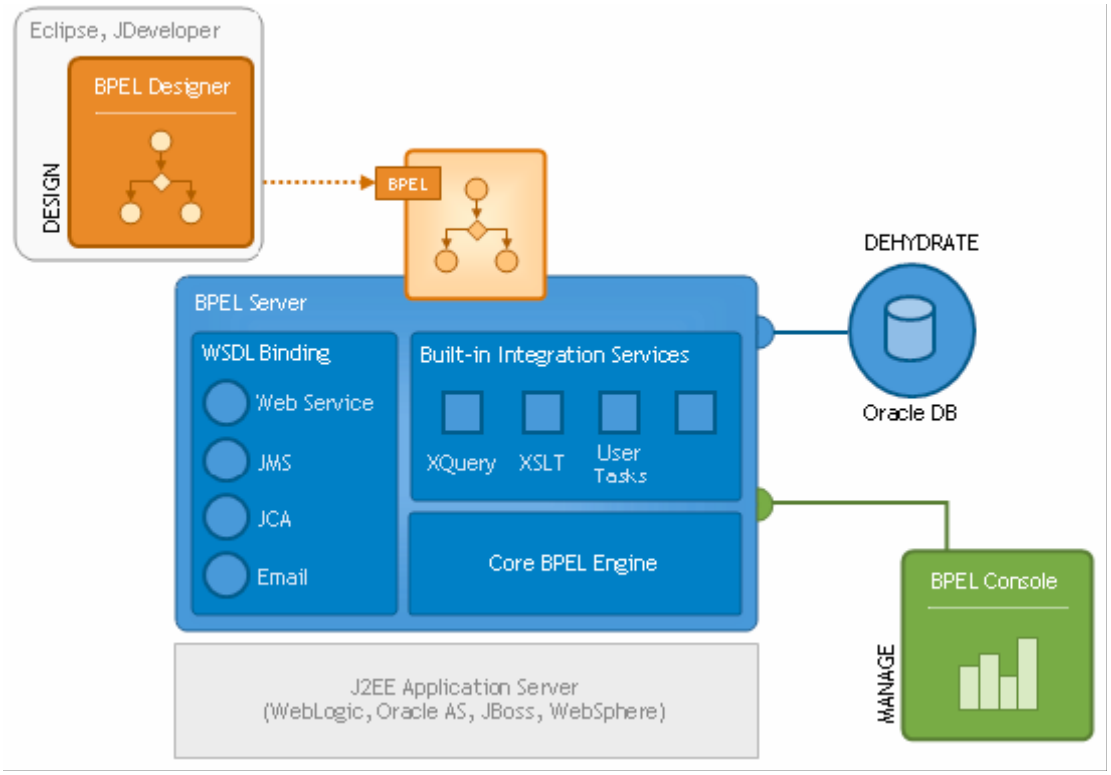

Figura 5.1 – Arquitetura do Oracle BPEL Process Manager

O BPEL Designer é a ferramenta de Construção dos workflows; o Core BPEL Engine é o núcleo de execução do workflow; os serviços "Built-in" facilita a conectividade e as capacidades de transformação de dados dos processos BPEL; a infra-estrutura "WSDL binding" permite a conectividade com outros protocolos além do SOAP; e por fim a console BPEL que permite a gerência, a administração e a depuração dos processos e workflows disponibilizados no servidor.

IBM WebSphere Process Server

O IBM WebSphere Process Server(WebSphere, 2008) é construído em cima da infra-estrutura SOA da IBM, desta forma incorpora todas as características padrões para integrar aplicações e serviços. Destacam-se na implementação do engine de execução de workflow:

- Suporte a serviços Web sobre diversos protocolos: SOAP/http e SOAP/JMS;
- Suporte a WSDL 1.1 e aos padrões de segurança e transações, WS-Security e WS-Atomic Transactions;
- Suporte aos protocolos de mensagens: JMS, WebSphere MQ e permitindo a comunicação multicast;
- Inclui pacotes para construções de clientes em C/C++ e .NET além de Java.

O IBM WebSphere Process Server encontra-se na sua versão 6.0, e o seu engine possui como destaque a portabilidade de instalação em diversos sistemas operacionais. Este pode ser instalado servidores com SO Microsoft 2000/2003; Unix em diversos sabores como Linux e AIX; e z/OS em Main-frames.

A seguir apresentamos a arquitetura do WebSphere Process Server:

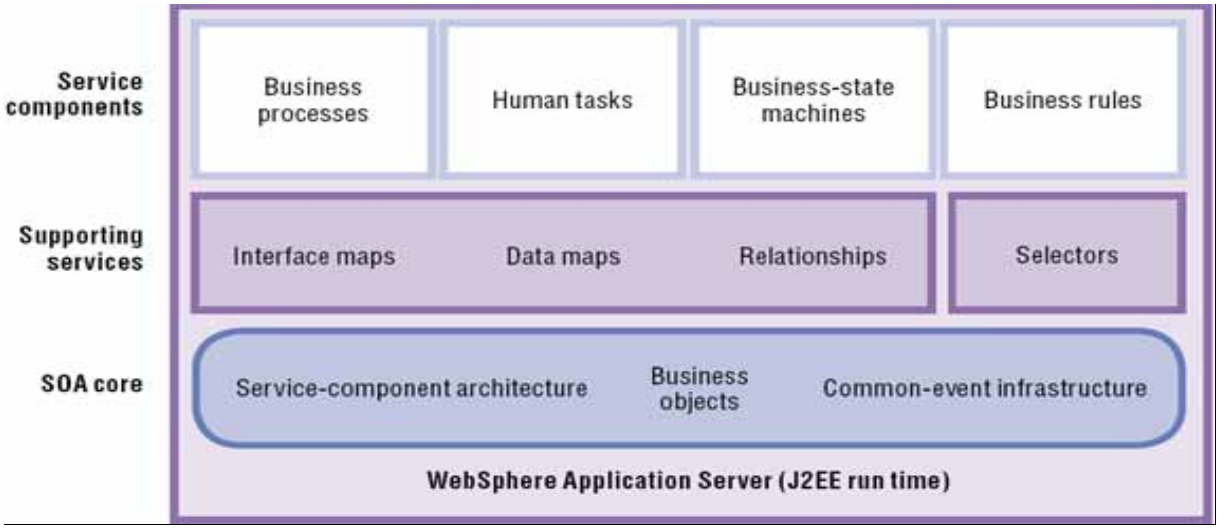

Figura 5.2 – Arquitetura do IBM WebSphere Process Server

Junto com o engine, o IBM WebSphere integra um editor de processos visual para a definição dos workflows. Este editor incorpora um depurador para acompanhar a execução passo-a-passo de um workflow BPEL. Como pode ser observado na arquitetura do WebSphere Process Server expande as capacidades do especificação do BPEL ao permitir a interação humana nos processos em execução, além disso, estes processos podem ser interrompidos e persistidos.

Microsoft BizTalk

As capacidades de orquestração do BizTalk(2008) são focadas na comunicação entre sistemas, suportando processos de negócio que dependem da integração de diversos softwares. Possui como principais focos a integração de arquitetura corporativas (EAI), B2B e gerência de processos de negócio (BPM).

O suporte a serviços Web do BizTalk é provido pela arquitetura ASP.NET que se beneficia da infra-estrutura Microsoft para a construção de workflows em conformidade com a WCF (Windows Communication Foundation) para a criação se aplicações orientadas a serviço. Além disso, suporta especificações padrão da indústria como: WS-Security, WS-ReliableMessaging e WS-AtomicTransaction.

A seguir apresentamos a arquitetura do BizTalk engine:

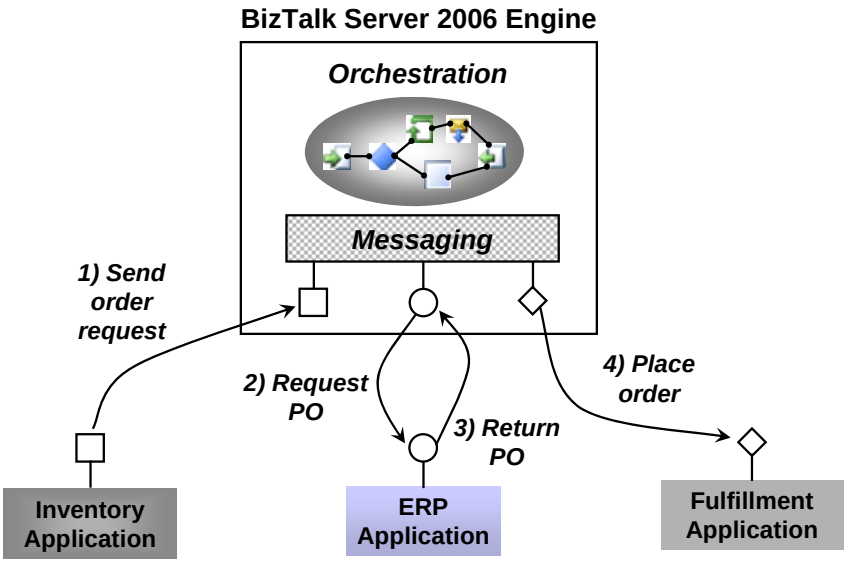

Figura 5.3 – Arquitetura do BizTalk Server 2006

O engine do BizTalk é calcado em duas características básicas: uma ferramenta para especificar e implementar a lógica do workflow e diversos mecanismos para a comunicação entre as aplicações participantes. No exemplo acima são explicitadas as diferenças de protocolos de comunicação entre os diversos atores do workflow.

Para implementar esta diversidade de interfaces de comunicação o BizTalk define um conjunto de adaptadores (adapters):

- Adaptador de Serviço Web – permitindo a comunicação SOAP/http;
- Adaptador de Arquivo – permite a leitura e escrita de arquivos no sistema de arquivo Windows;
- Adaptador http – permite o envio e recebimento de informação usando o http; e
- Adaptador MSMQ – para a comunicação utilizando o Microsoft Message Queuing;
- Adaptador WebSphere MQ – para a comunicação como o IBM WebSphere Message Queuing;
- Adaptadores SMTP – permitindo o uso de mensagens usando o SMTP;
- Adaptador POP – habilitando o recebimento e envio de mensagens através do POP3;
- Adaptador SQL – permitindo a leitura e escrita em um banco de dados Microsoft SQL Server.

Ao contrário dos outros engine BPEL o BizTalk não utiliza a infra-estrutura J2EE como suporte ao Engine, o que limita sobre maneira nas interfaces de comunicação da ferramenta.

JBoss jBPM

O JBoss jBPM (2008) é uma ferramenta para a construção de workflows gratuita que utiliza a infra-estrutura J2EE. Apesar do engine ser suportado por qualquer infra-estrutura J2EE, mas possui uma integração natural com o JBoss.

O JBoss utiliza uma linguagem própria para a construção de workflows, a jPdl – JBoss jBPM Process Definition Language. A jPdl se propõe a ser uma linguagem de especificação de processos flexível o suficiente para lidar com pequenos workflows departamentais até workflows corporativos e inter-corporativos. Para atender as ligações entre corporações a linguagem BPEL pode ser utilizada.

A seguir apresentamos a arquitetura do Engine jBPM:

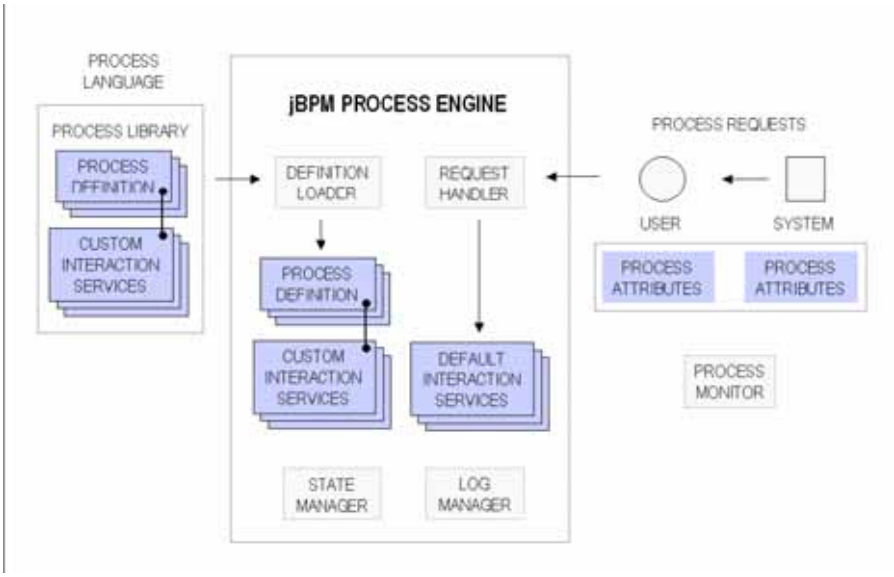

Figura 5.4 – Arquitetura do JBoss jBPM

O engine de processo mantem o controle dos estados e das variáveis de todos os processos ativos. O engine de processo é composto de um manipulador de solicitações (requeste handler) que provê a infra-estrutura de comunicação que aloca as tarefas para os processos adequados. Os serviços de interação (Interaction Service) expõem as aplicações existentes como funções ou dados nos processos finais. O gerente de estados (state manager) manipula os processos em execução no engine alocando os registros e dados além de preparar os acessos as bases de dados para as ações resultantes dos processos.

O monitor de processo (monitor process) provê a visibilidade dos estados finais dos processos em que os usuários e processos estão interagindo. A linguagem de processos é o núcleo do engine e se baseia em grafos diretos. Sobre este núcleo do engine de manipulação de grafos outros padrões são suportados: BPEL, BPELJ, BPML, ebXMLs, BPSS e WfMC's XPDL.

ActiveBPEL

O ActiveBPEL (2008) encontra-se na versão 2.0 em um modelo de distribuição gratuito. Os principais benefícios associados ao ActiveBPEL inclui: completude, pois implementa toda a especificação BPEL 1.1; força da industria, pois suporta características de ponta como persistência de processos, notificação de eventos e console das APIs; e trilha de crescimento, o modelo de desenvolvimento gratuito permite a evolução rápida através das contribuições da comunidade.

A seguir apresentamos a arquitetura do engine do ActiveBPEL:

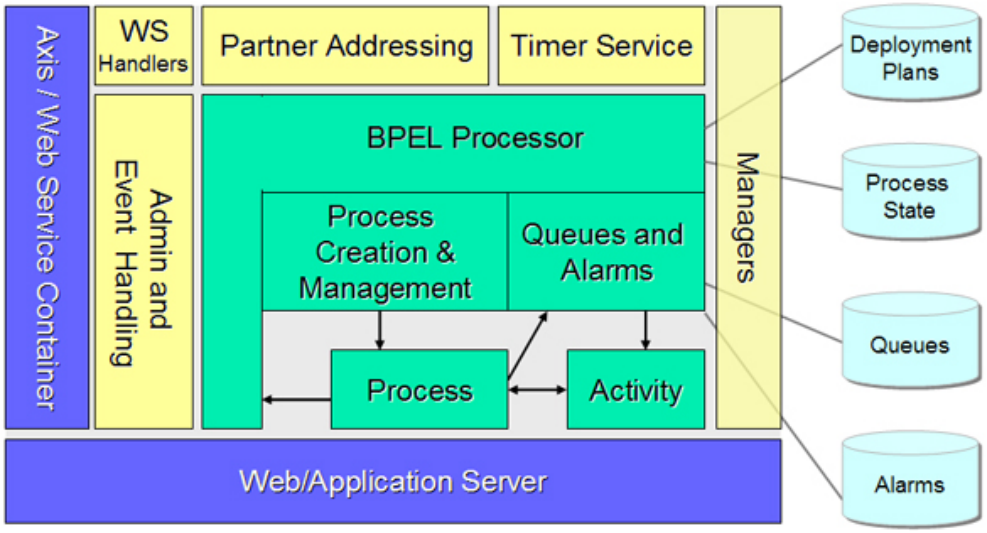

Figura 5.5 – Arquitetura do ActiveBPEL

O engine do ActiveBPEL roda sobre um container de servlet padrão tal como o Tomcat. Isto significa que os servidores de aplicações J2EE podem executar o ActiveBPEL. O engine também acompanha uma console de administração que permite redefinir parâmetros do engine assim como verificar informações dos workflows disponíveis. As bases de dados de persitência dos processos BPEL são implementados por um SGBD em memória que acompanha o ActiveBPEL, existe a intensão e incorporar o Tamino como SGBD XML para a arquitetura. Mas a característica que se destaca no engine é a possibilidade de implantação de workflows a quente.

5.2 - myGrid

O myGrid é um projeto financiado pelo EPSRC (*Engineering and Physical Sciences Research Council*), envolvendo cinco universidades do Reino Unido, o grupo *European Bioinformatics* e alguns colaboradores da indústria.

Este projeto tem como principal objetivo explorar a tecnologia de computação em Grid e prover camadas intermediárias úteis às necessidades da bioinformática. Para isso, foram construídos serviços para integração de dados e recursos de aplicações, como descoberta de recursos, execução de workflows e processamento distribuído de consultas. Além disso, também possibilita a gerência de dados de proveniência, metadados e personalização, o que permite integração e um modelo semântico para a área de bioinformática (myGrid, 2008).

Dessa maneira, o myGrid, oferece suporte à definição de experimentos biológicos *in-silico*, modelados como workflows em um ambiente de Grid (Zhao, 2004).

É um software de código aberto orientado a serviços, implementado em Java e scripts shell, podendo ser utilizado em plataformas Windows ou Linux. Os workflows são definidos na linguagem *X-Scufl* através de uma interface gráfica disponibilizada no ambiente Taverna Workbench, um dos módulos do myGrid (Zhao, 2004).

Nas subseções abaixo, faremos uma breve apresentação da linguagem de definição de workflows do myGrid, assim como da sua máquina de execução de workflows, Freefluo. Também estaremos abordando a gerência de dados no myGrid e seu ambiente para definição e execução de workflows científicos, o Taverna Workbench.

5.2.1 – Scufl

A Scufl (*Simple Conceptual Unified Flow Language*) é a linguagem para definição de workflows, originalmente baseada na WSFL, desenvolvida pela equipe do myGrid para prover um nível mais alto de abstração relacionado à visão dos workflows definidos.

O objetivo foi tornar a tarefa de definição de workflows bem mais simples, permitindo ao usuário mapear tarefas conceituais em entidades simples com o mínimo de esforço possível.

A Scufl possui três entidades principais definidas, que podem ser representadas por tags na linguagem. São elas: *processors* (define o que será executado em um determinado passo do workflow), *data-links* (define uma ligação entre portas de entrada e saída dos processors) e *coordinations* (define a coordenação entre processors).

Esta linguagem também fornece recursos básicos para facilitar a tolerância à falhas e a captura de dados de proveniência.

Durante a execução de uma etapa do workflow, serviços web envolvidos podem estar indisponíveis, tornando muito importantes os mecanismos como recursos para configuração de timeouts e número de tentativas para reexecução, além da definição de processadores alternativos para o caso de falha.

Estes mecanismos são representados por metadados expressos na linguagem e a inclusão destes recursos foram algumas das motivações para o desenvolvimento de uma nova linguagem, própria para o myGrid. A desvantagem é que por ser uma solução proprietária, atualmente só pode ser executada pelo Freefluo.

No contexto científico, os workflows têm seu valor reduzido caso outros cientistas não sejam capazes de verificar a origem ou proveniência do experimento representado. Dados que representem essa proveniência são tipos de metadados essenciais para que os experimentos sejam validados e verificados. Eles representam o propósito do experimento, seus resultados ou anotações mantidas pelo cientista, além de dados sobre a qualidade do experimento ou seus dados temporais.

Na Scufl são representados dois tipos básicos de metadados relacionados à proveniência do experimento: *Annotations* (anotações padrão como, "última atualização" ou "que é o dono" do experimento) e *Derivation Path* (dados relacionados com a execução do workflow, resultados obtidos a partir dos dados de entrada).

5.2.2 – Freefluo

O Freefluo é uma máquina de execução de workflows, criada pelos mesmos desenvolvedores do myGrid, sendo uma solução específica para este ambiente.

Ele permite o gerenciamento da execução de workflows para Web Services, pois suporta WSDL para invocar serviços. Além disso, controla as tarefas de escalonamento dos serviços, gerenciamento de eventos seqüenciais ou paralelos, assim como, tolerância à falhas (Freefluo – Sourceforge, 2008).

Oferece suporte a linguagens baseadas na WSFL, mas como funcionalidade mais específica do myGrid, também processa as suas linguagens Scufl e X-Scufl. Além disso, é facilmente extensível para oferecer suporte a diferentes métodos de invocação, como Grid Services ou CORBA (Addis et al., 2003).

Aceita definições de wokflows e dados de entrada do Taverna, mas não está preso a esta arquitetura pois é possível executar workflows definidos em WSFL, ou na Scufl.

O workflow, para o Freefluo, é um objeto interno representando o modelo de workflow como um grafo direcionado, no qual cada nó possui uma máquina de estado que define seu ciclo de vida. A sincronização e a transição de estados são controladas pela transmissão de mensagens entre os nós, conforme o progresso da execução. Possui um parser de linguagem de workflow, para convertes uma especificação textual de workflow para uma representação interna do objeto. A execução ocorre com a invocação de serviços, através de chamadas WSDL, em tempo de execução. Neste momento ocorre também o tratamento dos tipos de dados específicos, passados entre os serviços, através de mensagens XML (myGrid, 2008).

5.2.3 – Gerência de dados no myGrid

Inicialmente o myGrid realizava a captura e o armazenamento da proveniência através de um serviço *web* chamado MIR que também definia um esquema de dados relacionados ao domínio da bioinformática extremamente simples, contendo os dados de execução do *workflow* e seus dados intermediários, de entrada e saída, sem armazenar esses dados intermediários em um esquema mais apropriado para o domínio da biologia. Essa solução restringia muito o uso de outros esquemas de dados (myGrid, 2008).

A nova versão do Taverna conta com uma outra alternativa para a gerência dos dados de proveniência, o *Taverna Provenance Plugin* (Taverna Provenance Plugin, 2008). Com isso, o controle da proveniência, assim como seu nível de detalhe passou a ser opcional para seus usuários. Os *logs* são armazenados durante a execução do *workflow*. Essa nova estratégia mapeia os metadados relacionados à execução do *workflow* para uma ontologia definida pela equipe do Taverna e, em seguida, estes metadados são armazenados em uma base de dados. Os dados intermediários gerados para cada instância executada do *workflow* não são classificados segundo uma ontologia e são armazenados apenas como objetos e relacionamentos entre eles. Essa solução é mais flexível, porém encontra-se ainda em estágio inicial de desenvolvimento.

O myGrid também utiliza o repositório KAVE (*Knowledge Annotations and Verification of Experiments*) para armazenamento do conhecimento semântico dos experimentos científicos da bioinformática. O KAVE possibilita a outros componentes do myGrid armazenar e buscar dados de anotações e verificações. Estes dados são expressos em RDF (*Resource Description Framework*) e consultados usando a RDQL (*Resource Description Query Language*). Os dados, como, por exemplo, um relacionamento significativo entre a saída de um serviço e a entrada de outro, são capturados e armazenados durante a execução do workflow. Os dados armazenados no KAVE estão altamente envolvidos com as informações de proveniência armazenadas pelo Taverna Provenance Plugin.

5.2.4 – Taverna Workbench

O Taverna Workbench é um dos componentes do myGrid. Esta ferramenta permite que os usuários possam construir workflows complexos, executar estes workflows e visualizar seus resultados (myGrid, 2008).

O Taverna possui uma interface gráfica por meio da qual é possível construir, executar e alterar workflows, facilitando as atividades de bioinformática com a execução dos experimentos *in-silico* representados pelos workflows.

Este ambiente é integrado com a linguagem Scufl e com a máquina de execução Freefluo, como pode ser observado na figura 5.6. Também é composto por quarto módulos principais: o *Advanced Model Explorer* (*Scufl Model Explorer*), o *Workflow Diagram* (*Scufl Diagram*), *Avaiable Services* e o *Enactor Launch Panel* (Oinn et al., 2004).

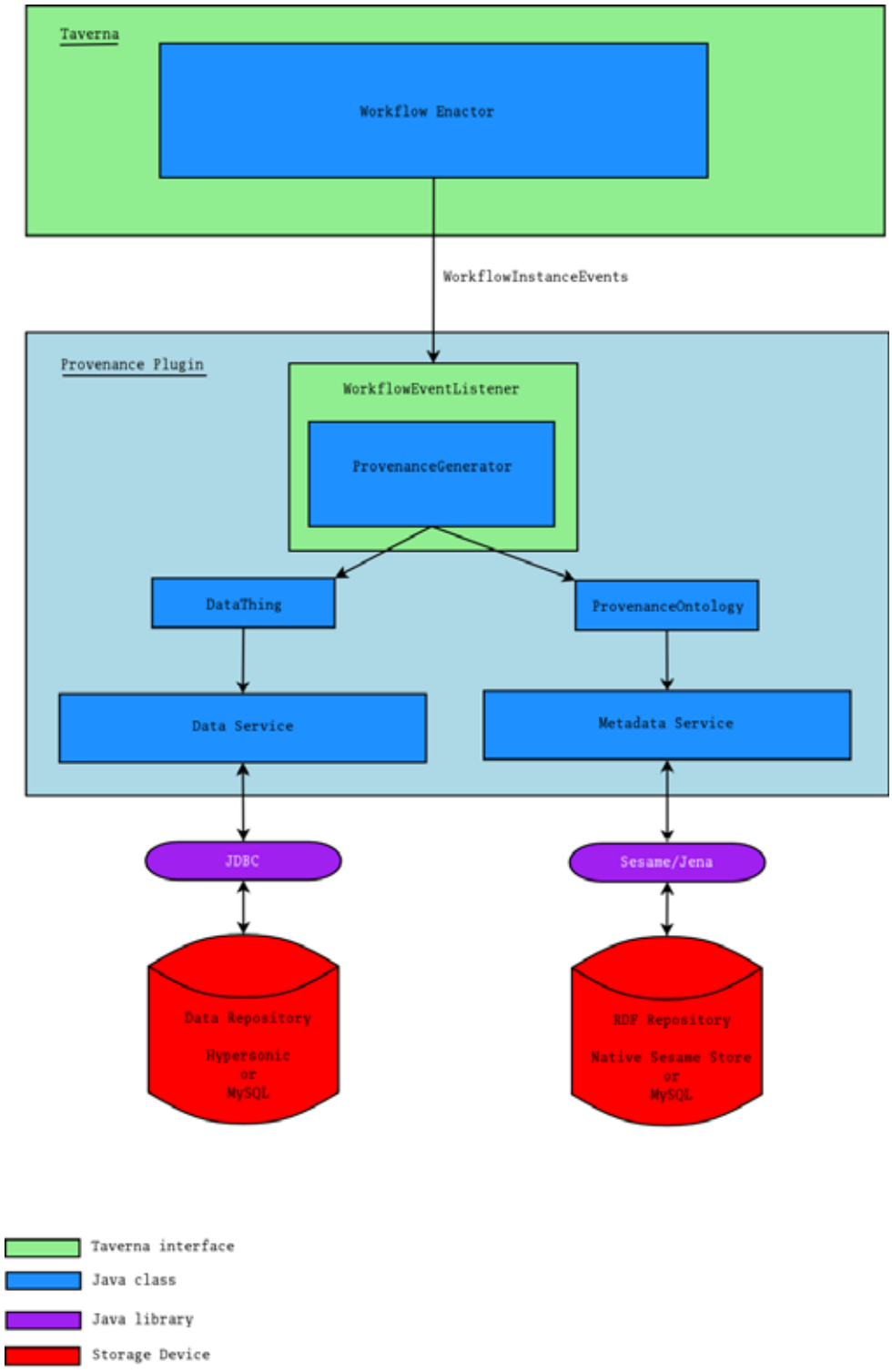

Figura 5.6 – Arquitetura do Taverna Workbench (Taverna, 2008)

O módulo *Advanced Model Explorer*, onde a construção visual do workflow e suas configurações permite que um usuário sem conhecimentos da linguagem Scufl possa definir um workflow facilmente, apenas escolhendo os serviços a serem executados, configurando seus parâmetros de entrada, assim como a seqüência das atividades.

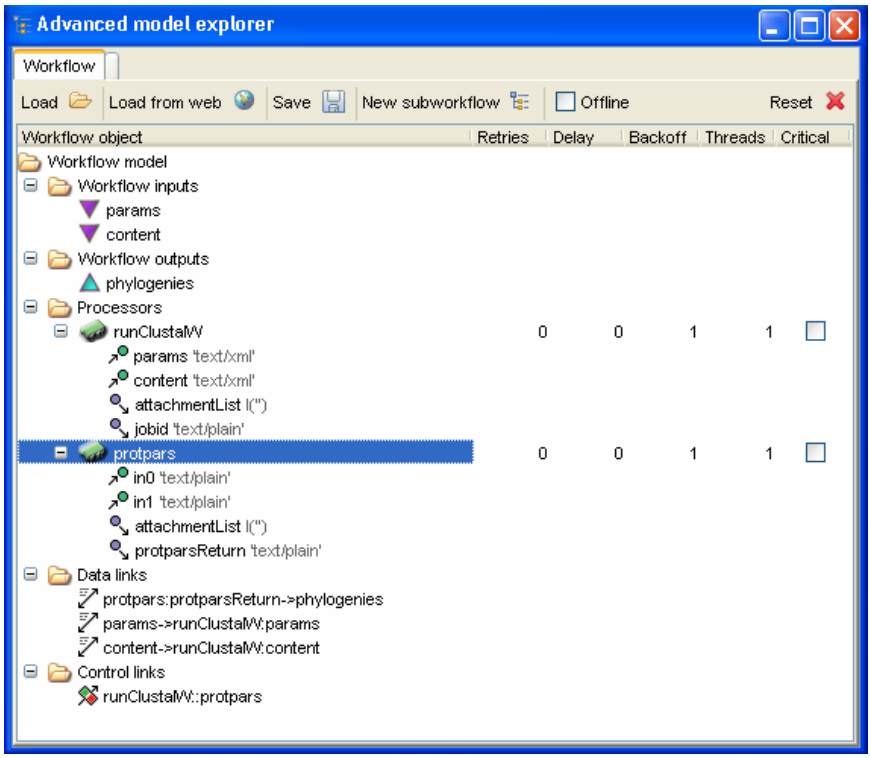

Figura 5.7 – Advanced Model Explorer (myGrid, 2008)

O módulo *Workflow Diagram* permite uma representação gráfica do workflow que está sendo definido. Não é permitido editar o workflow a partir deste módulo, cujo principal objetivo é facilitar a visualização do experimento *in-silico* representado. Neste módulo é possível ajustar o gráfico desenhado ao tamanho da janela, exibir portas e tipos de dados, escolher o modo de desenho (horizontal ou vertical) e salvar a imagem como figura.

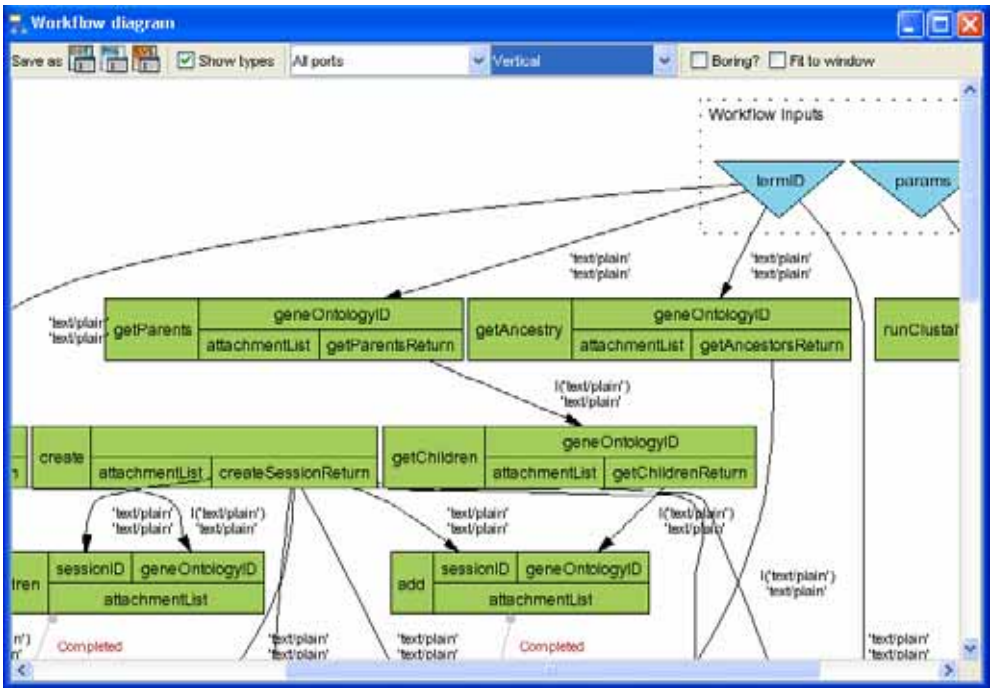

Figura 5.8 – Workflow Diagram (myGrid, 2008)

O módulo *Avaiable Services* é responsável pela exibição de serviços disponíveis. É neste módulo que o usuário selecionará os serviços que serão executados em cada *processor*. Estes serviços podem ser locais ou remotos. Além disso, o usuário pode incluir ou retirar um serviço da lista de serviços disponíveis.

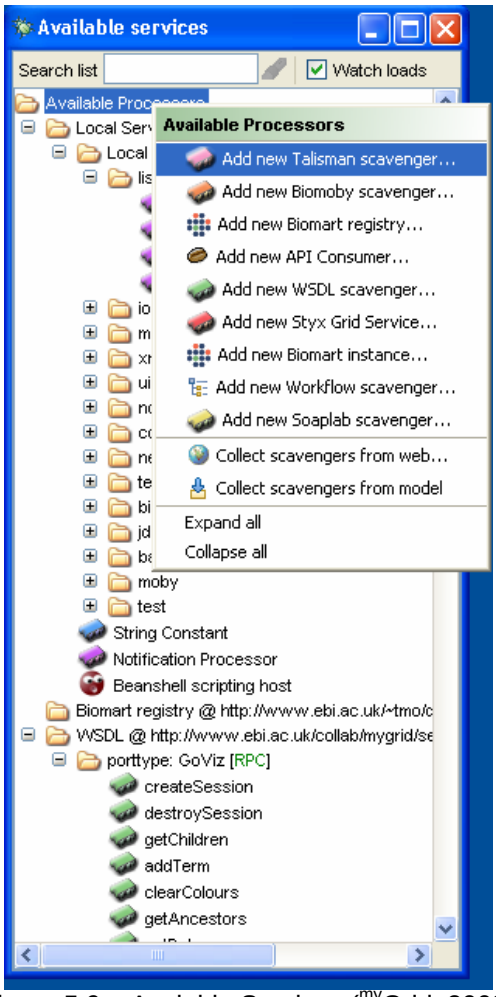

Figura 5.9 – Avaiable Services (myGrid, 2008)

O módulo Enactor Launch Panel, representa o estado atual da execução de um workflow. Nele é possível visualizar o passo que está sendo executado, se houve falha ao executar alguma etapa, além de apresentar o resultado final da execução do workflow e seus resultados intermediários.

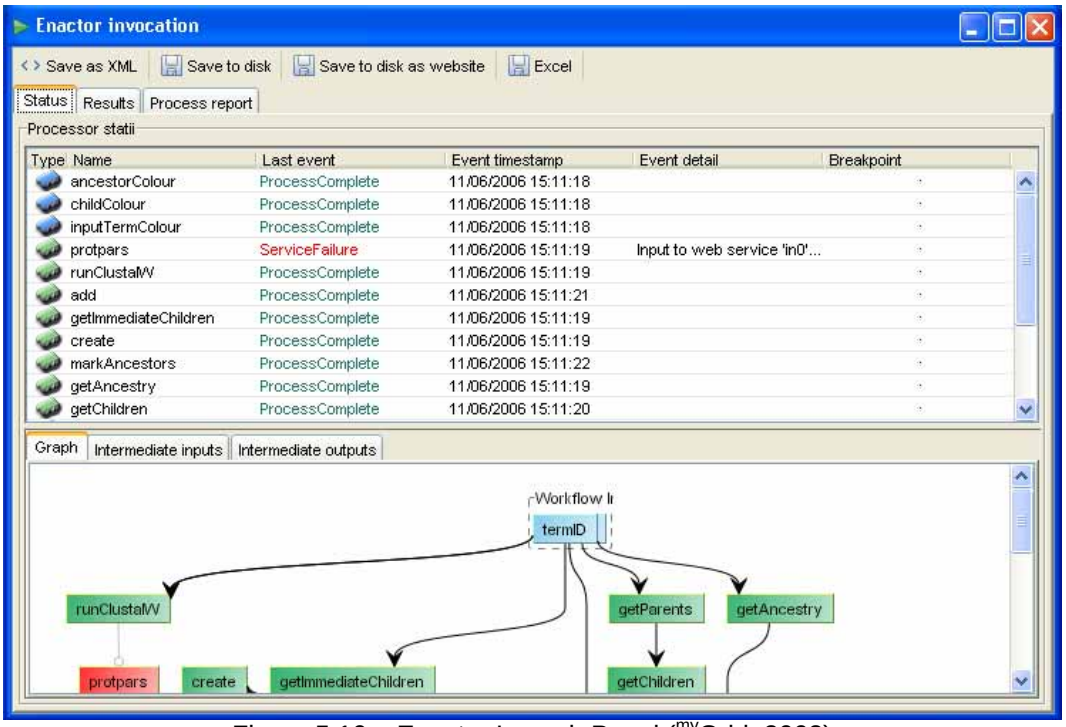

Figura 5.10 – Enactor Launch Panel ([my]Grid, 2008)

5.3 - O Projeto PtolemyII - Kepler

O Kepler é a máquina de execução de workflows do projeto Ptolomy II. Ele é um sistema de código aberto que tem como objetivo atender múltiplos domínios do conhecimento, dentre eles destacamos, a Geologia, Física, Química e Bioinformática. Ele tem como meta desenvolver soluções genéricas tanto para o processamento de workflows científicos quanto para os desafios de integração de aplicações científicas (Ludäscher et al, 2006), além disso, ele obedece aos requisitos estabelecidos por McPhillips e Bowers (2005) e McPhillips et al (2006). Nesta seção faremos um contextualização histórica do projeto PtolemyII-Kepler e, em seguida a seguida apresentamos as principais características do arcabouço. Finalizamos a seção apresentando uma visão crítica do produto.

5.3.1 - Um breve histórico do projeto

O projeto Ptolemy iniciou-se a partir de um grupo de pesquisadores da U.C. Berkeley. Este projeto é composto por três gerações, e no momento ele se encontra na terceira. A primeira geração (1986-1991) produziu um software chamado Gabriel, ele foi escrito em LISP e tinha como objetivo executar o processamento de sinais. Durante seu desenvolvimento, surgiu a necessidade de se trabalhar com fluxos de dados síncronos (SDF) e fazer uso de técnicas de escalonamento seqüencial e paralelo. Gabriel possuía uma interface gráfica primária baseada na ambiente gráfico Vem e fazia uso do banco de dados Oct, ambos eram projetos da Universidade de Berkeley.

A segunda geração (1990-1997), denominada Ptolemy Classic, surgiu no início dos anos 90, e envolveu uma recodificação para a linguagem C++. Ele foi o primeiro ambiente modelado para suportar inúmeros modelos de computação, como por

exemplo: fluxo de dados booleanos (BDF), fluxos de dados dinâmicos (DDF), fluxos de dados multidimensionais (MDSDF) e redes de processos (PN). Nesta geração surgiu o conceito de componentes simples e componentes "*high-order*" (usados para encapsular componentes simples). A interface gráfica do Ptolemy Classic também foi baseada no Vem e Oct, entretanto ela foi expandida pelo arcabouço Tycho. Alguns módulos desta versão foram comercializados como parte dos sistemas Agilent ADS (AGILENT, 2008) e do Cadence SPW (Bhattacharyya, 2005a).

A geração atual, agora denominada Ptolemy II, mais uma vez foi reescrita, desta vez sob a forma de um arcabouço de software na linguagem Java. Inúmeras novidades foram adicionadas, dentre elas destacamos: a possibilidade de modelar e executar workflows associados a domínios polimórficos (os componentes do arcabouço podem ser utilizados em múltiplos domínios); novos modelos modais (máquinas de estados finitos são combinadas hierarquicamente como outros modelos de computação); a elevada modularidade, a versão atual é composta por packages que podem ser usados de forma independente; separação entre os modelos *abstratos* (diagramas) e *concreto* (implementação) do workflow. A versão atual da máquina de workflow chama-se Kepler. Ela herdou uma série de características das gerações anteriores, porém a versão atual faz uso de um sofisticado modelo de classes, capaz de representar e operar sobre inúmeros tipos de dados heterogêneos, além disso, é possível invocar serviços Web e serviços de grid. Uma nova interface gráfica em Java, denominada Vergil, foi associada ao projeto. As principais novidades residem no conceito de classes de componentes (*actors)* e nos modelos de computação (*directors*). Os primeiros promovem um elevado grau de independência, uma vez que desacoplam a semântica do workflow da implementação das classes, enquanto que os demais são encarregados da execução do e gerenciamento do ciclo de vida do workflow segundo um determinado modelo de computação. A figura 5.12 apresenta a nova interface do PtolemyII.

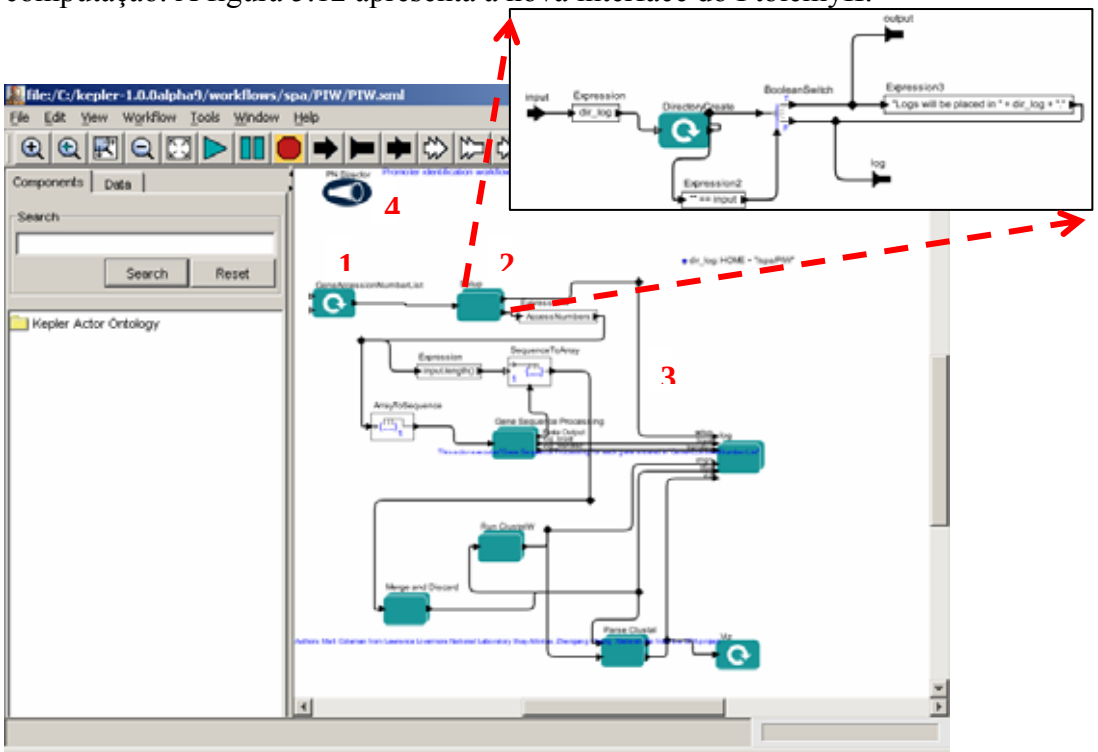

Figura 5.12. – A figura ilustra a interface do Kepler com um workflow, onde é possível identificar um ator simples (1), um ator composto(2), uma indicação de canal de dados (3) e um diretor (4).

Atualmente, o projeto Ptolemy está passando por mais uma ampliação. O grupo de pesquisa da Universidade de Berkeley acaba de lançar uma nova versão da máquina de execução de workflows científicos (Kepler-RC1), com previsão de lançamento da versao 1.o em Março de 2008.

5.3.2 - Modelagem e Design de workflows

Uma simulação/execução no PtolemyIKepler não é uma exclusividade da Bioinformática, pelo contrário, ele é bastante utilizado para fazer modelagens e simulações de sistemas concorrentes. Um dos seus principais focos são os sistemas de software embutido (Lee, 2000), particularmente aqueles sistemas que mesclam tecnologias distintas, como por exemplo, projetos que utilizam componentes eletrônicos digitais e analógicos, software e hardware, dispositivos eletrônicos e digitais, entre outros. A ferramenta também ataca os problemas relacionados com modelos de operações complexas, tais como, processamento de sinais, controle de feedback, decisões seqüenciais, entre outros. Conceitualmente o Kepler distingue a modelagem do design de workflows.

A *modelagem* é uma ação, desenvolvida pelo usuário, que representa formalmente um sistema ou subsistema. Um modelo de sistema pode ser de dois tipos: matemático ou construtivo. O modelo matemático é representado sob a forma de equações e propriedades, já os modelos construtivos são definidos como um conjunto de procedimentos computacionais que mimetizam as propriedades de um sistema, normalmente este tipo de modelo são utilizados para descrever o comportamento de um sistema em resposta a estímulos externos. Os modelos do tipo construtivo também são chamados de modelos executáveis ou simulações.

O *design* é o ato que define o workflow propriamente dito. Geralmente, ele envolve a definição e refinamentos de um ou mais modelos de sistemas até que alcancem à funcionalidade desejada. Para o Kepler, a modelagem e o design são duas atividades distintas, porém intimamente relacionadas. Por exemplo, em alguns casos os modelos são imutáveis, enquanto que o design não é. Isto é, suponha que um pesquisador disponha de um equipamento mecânico (um motor qualquer), é possível modelar este sistema mecânico (pré-existente) e fazer o design dos sistemas eletrônicos e simular seu comportamento naquele dispositivo físico.

5.3.3 - Modelos de computação do Kepler e softwares embutidos

O projeto PtolemyII-Kepler possui uma característica muito interessante, mesmo tendo evoluído para um arcabouço que oferece amplo suporte para o design e execução de workflows científicos, ele ainda oferece amplo suporte para modelagem de softwares embutidos. Software embutido é um software que reside um dispositivo físico qualquer (não necessariamente um computador). Ele é pervasivo, isto é, está totalmente integrado ao ambiente do usuário e objetiva auxiliá-lo em suas tarefas diárias. Este ambiente é altamente dinâmico e heterogêneo, portanto, requer modelos de computação apropriados (Augustin, Lima, Yamin, 2006).

Os modelos executáveis (simulações) no Kepler são construídos de acordo com um determinado modelo de computação que obedece a um conjunto de leis da Física. Por exemplo, para um modelo executável que descreve um dispositivo eletrônico qualquer, o modelo de computação representará as leis da Física aplicáveis ao contexto, isto é, a representação de grandezas, a passagem do tempo, a concorrência entre eventos, entre outras.

O Kepler possui diversos modelos de computação, cada modelo trata a concorrência e o tempo de forma diferenciada, portanto, cada modelo de computação pode representar uma semântica distinta para os mesmos componentes do workflow de um determinado domínio. Dentre os principais modelos de computação destacamos:

- **Component Interaction (CI)** – Este modelo representa sistemas do tipo data-driven (uma aplicação ou serviço Web utiliza um serviço de cotação de ações provido por terceiros) ou demand-driven (usuário utiliza interativamente um serviço comércio eletrônico). Os modelos CI são utilizados para simular e estudar como os softwares embutidos manipulam eventos assíncronos, tais como interrupções externas e entradas/saídas de dados assíncronas.

- **Communicating Sequential Processes (CSP)** – É um modelo não determinístico que representa sistemas que possuem processos concorrentes e que se comunicam através da troca de mensagens atômicas e síncronas. Este modelo de computação também é conhecido como *rendezvous* e se aplica a sistemas que compartilham algum tipo de recurso, como por exemplo, sistemas cliente-servidor, ou hardwares que utilizam recursos de multiplexação ou multitarefa.

- **Continuous Time (CT)** – Nesse modelo os atores interagem através de sinais contínuos (circuitos analógicos ou mecânicos). Os atores geralmente especificam relações algébricas ou diferencias entre os dados de entrada e saída. Este modelo é especialmente interessante para representar sistemas que fazem uso de equações algébricas/diferencias não lineares.

- **Discrete-Events (DE)** – Este modelo permite que os atores utilizem um relógio único e se comuniquem através de seqüências de eventos temporais, onde cada evento consiste de um valor e um timestamp. Este modelo é especialmente útil para simulações de circuitos digitais e sistemas de telecomunicações.

- **Dynamic Data Flow (DDF)** – Este modelo é um super set dos domínios synchronous dataflow (SDF) and Boolean dataflow (BDF). No domínio SDF, um ator consome e produz um número fixo de tokens por execução. Essas informações são estáticas e permitem que o escalonamento de serviços seja definido em tempo de compilação. No modelo DDF, um ator é capaz de alterar as taxas de produção e consumo de tokens por execução, logo não é possível definir o escalonamento dos serviços em tempo de compilação.

- **Discrete Time (DT)** – Este modelo estende o domínio SDF, ele introduz a noção de intervalos de tempo entre os tokens. A comunicação entre os atores dá-se sob a forma de uma seqüência de tokens que podem ser disparados em intervalos de tempo regular ou não.

- **Finite State Machine (FSM)** – Este modelo de computação é radicalmente diferente dos demais, ele não está baseado em atores, mas em estados e, as conexões entre os atores representam transições entre estados. Os modelos FSM são muito utilizados para expressar lógicas de controle e para construir modelos modais.

- **Process Network (PN)** – Neste modelo os processos trocam mensagens através de um canal que é capaz de buferizá-las. O remetente de uma mensagem não precisa aguardar pelo receptor para enviar uma nova mensagem. Este estilo de comunicação também é conhecido como passagem de mensagens assíncrona.

Os modelos de computação são representados sob a forma de componentes especiais denominados diretores (*directors*). Os directors são encarregados de separar as questões de orquestração das de execução de um workflow. Atualmente nem todos os modelos de computação estão representados no Kepler, existem apenas quatro tipos de *directors*, a saber: Synchronous Dataflow (SDF) , Process Network (PN), Continuous Time (CT), Discrete Event (DE). Maiores detalhes sobre os diretores serão apresentadas nas próximas seções.

5.3.4 - A arquitetura do PtolemyII-Kepler

O PtolemyII-Kepler é uma arquitetura modular e escalável baseada na tecnologia Java. A arquitetura oferece uma infra-estrutura unificada que permite a implementação de um grande número de modelos de computação (Figura 5.13). A arquitetura é composta por um grande número de pacotes genéricos (aqueles que implementam os modelos de computação e atores) e pacotes específicos (aqueles que contém funções específicas, como por exemplo bibliotecas matemáticas, grafos, funções multimídia, entre outras). Segundo Ludäscher (2003, 2005, 2006) a versão atual do Kepler foi construída sobre o arcabouço PtolemyII e já oferece suporte total ao ciclo de vida de um workflow científico (seção 2.4), ele é capaz de executar tarefas com diferentes granularidades (de atores de elavada granularidade que encapsulam serviços Web até atores específicos para transformações de dados em simulações de circuitos eletrônicos). O Kepler também suporta um amplo leque de diferentes tipos de workflows, ele é capaz de executar desde pipelines locais até aplicações distribuídas de alto desempenho.

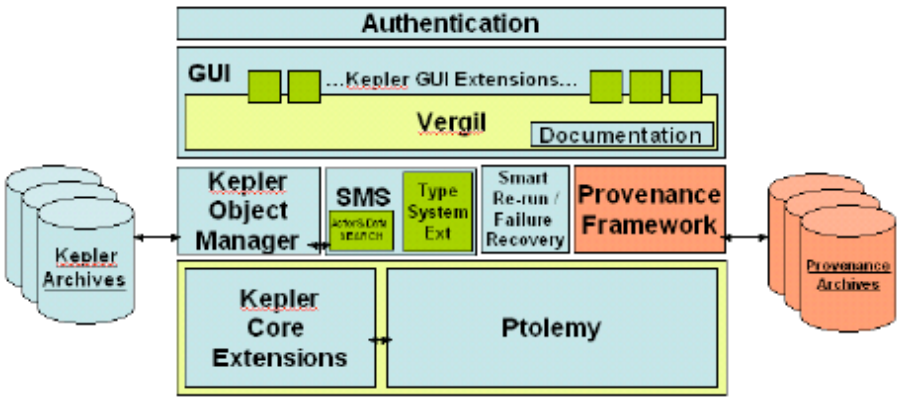

Figura 5.13. – Arquitetura do Kepler segundo Altintas, Barney e Jaeger-Frank (2006)

O Kepler Object Manager gerencia todos os pacotes (classes de atores e diretore) do ambiente. Bhattacharyya et al. (2005a), classificam os pacotes da arquitetura como sendo de quatro tipos: *core packages, UI packages*, *library packages* e *domain packages*. Os core packages suportam os modelos de dados, ou sintaxe abstrata dos diagramas definidos pelo usuário. Os UI packages oferecem suporte tanto para a manipulação de arquivos no formato XML (MoML) quanto uma interface visual que é utilizada na construção gráfica dos workflows. Os library packages são as bibliotecas utilizadas para definir os atores polimórficos, uma vez que eles são capazes de operar independentemente em uma grande variedade de domínios. Finalmente, as domain packages dizem respeito aos domínios de aplicação, cada uma deles pode implementar um único modelo de computação. A descrição de todos os pacotes e classes presentes no PtolemyII está fartamente documentada em Bhattacharyya et al. (2005a, 2005b e 2005c)

5.3.4.1 - Core packages

Os *core packages* formam a estrutura de classes central do PtolemyII. Via de regra, esses pacotes são dependentes de pacotes menores. Os pacotes definem estruturas dos wfs que podem ser representadas sob a forma de grafos do tipo cluster (*clustered graphs*). Os grafos fornecem a *sintaxe abstrata* (organização conceitual) necessária para manipular os diagramas de transição de estado, diagramas de bloco e netlists definidos pelo usuário. Os principais pacotes do core são: *kernel, data, actor, copernicus, graph, math, mathlab* e *util*.

5.3.4.2 - UI packages

Os pacotes da UI fornecem o suporte necessário à manipulação do ambiente gráfico e dos arquivos XML produzidos pelo Vergil. Os arquivos XML utilizam uma meta-linguagem denominada MoML (Modeling Markup Language) e armazenam localmente um modelo de interconexão dos componentes (atores e diretores) de um dado workflow. Dentre as principais características do MoML destacamos: (1) a facilidade de integração com a Web, (2) independência de aplicação, uma nova versão de Kepler que seja capaz de ler os arquivos MoML pode facilmente carregar todas as classes e pacotes associados a definição do workflow, (3) extensibilidade, os componentes (atores) podem ser parametrizados através das propriedades dos elementos XML ou mesmo através de arquivos externos, (4) independência de semântica, os arquivos MoML definem uma semântica relativa a interconexão dos componentes e parâmetros de um dado workflow. Os principais pacotes do UI são: *actor.gui, gui, media, moml* e *vergil*.

5.3.4.3 - library packages

A maioria dos domínios do PtolemyII estende as classes definidas no pacote *actor* para uma interpretação semântica mais apropriada. No pacote library estão definidas a maior parte dos atores, esses atores são definidos como atores polimórficos pois não estão associados a nenhum domínio. Os principais pacotes da library são*: actor.lib, actor.lib.comm, actor.lib.conversions, actor.lib.gui, actor.lib.hoc, actor.lib.image,*

actor.lib.io, actor.io.comm, actor.lib.jai, actor.lib.javasound, actor.lib.jmf, actor.lib.jxta, actor.lib.logic, actor.lib.net e actor.lib.python.

5.3.5 - Design de workflows no Kepler

Seguramente, uma das mais importantes características do PtolemyII-Kepler diz respeito ao paradigma de modelagem orientada a atores. Essa característica facilita as tarefas de design e execução de workflows científicos, pois separa dois conceitos distintos a comunicação (fluxo de dados) da coordenação do workflow (orquestração). No Kepler, um workflow pode ser compreendido como uma composição independente de componentes denominados atores (*actors*).

A habilidade de compôr serviços Web dinâmicamente é cada vez mais importante para a Bioinformática, ela permite que grupos de cientistas geograficamente distribuídos trabalhem em conjunto na captura, correlação, análise e compartilhamento de dados. Atualmente existem vários exemplos de SGWf científicos baseados em composição dinâmica, a saber: ^{my}Grid, Triana, Kepler. A definição de workflows no Kepler apresenta ambigüidades, por exemplo, eles ainda não são especificados formalmente, isto é, o design de um workflow no Kepler não está baseado em nenhuma das técnicas apresentadas na seção 4.1 Recentemente, Berkley et al (2005) apresentaram os primeiros trabalhos relacionados com este tópico, porém, essa característica não faz parte da documentação do produto.

5.3.5.1 - Ambiente de edição de workflows

O Kepler pode ser utilizado como ferramenta de simulação, editor de diagrama de blocos, sistema de prototipagem rápida, toolkit de desenvolvimento de componentes e aplicações Java. Entretanto, um de seus componentes mais versáteis é o Vergil, ele é o editor de diagramas de bloco, ele possibilita a criação e configuração de workflows científicos, além da manipulação direta durante sua execução. A interface visual do Vergil é bem simples, ela está divida em três áreas principais. A primeira (Figura 5.14) diz respeito ao menu de comandos e a barra de ferramentas, ela permite que o usuário amplie ou reduza a visualização de um workflow; execute, pare ou interrompa sua execução, ou ainda adicione portas de entrada/saída em um determinado ator composto (detalhe da figura 5.14).

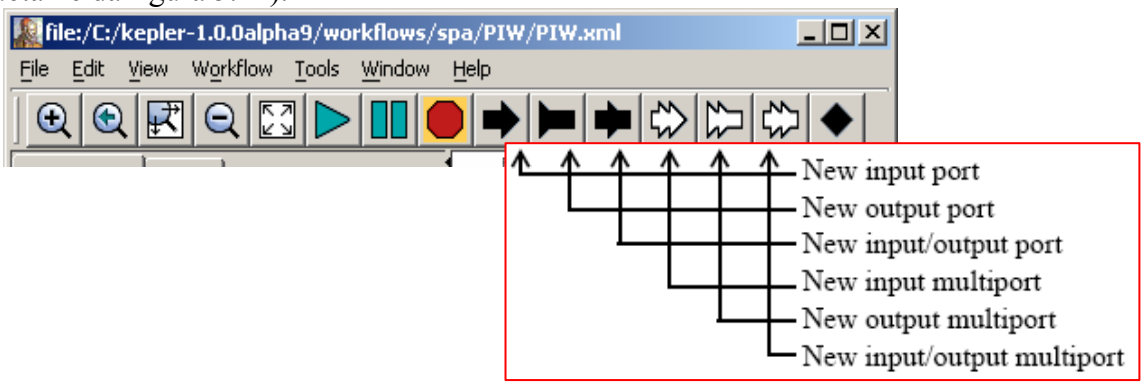

Figura 5.14. – Detalhe da barra de ferramentas do Vergil

A segunda área importante do Vergil é a paleta de componentes, representada na Figura 5.15. Os componentes são subdivididos dois grupos principais: *Directors* e *Workflow*

components. Esses estão subdivididos em *Actors*, *Parameter* e *Variable*. A adição de componentes a um dado workflow é uma tarefa relativamente simples, basta que o usuário selecione os componentes adequados e arraste-os para a área de edição de componentes (Figurax+1).

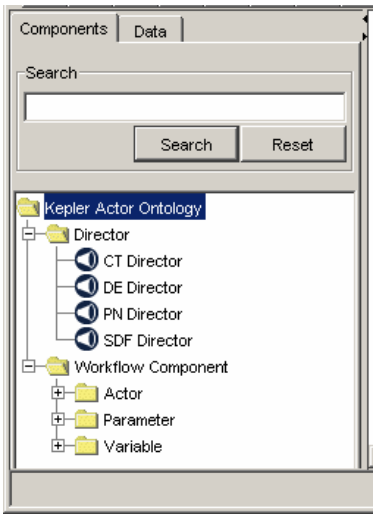

Figura 5.15. – Paleta contendo os diretores e componentes de um workflow

A interface visual Vergil apresenta outras funcionalidades bem interessantes, dentre elas destacamos*: a janela de Runtime* e o *menu de ferramentas do Vergil* (Figura 5.16). A janela Runtime permite que o usuário acompanhe a execução do workflow. Nela, o usuário pode visualizar/alterar parâmetros de entrada do workflow, visualizar dados que são produzidos, interromper e temporariamente a execução de um workflow e continuar a partir deste ponto. A janela não permite que se alterem os parâmetros iniciais do workflow que já está em execução, neste caso a execução corrente será finalizada e uma nova execução será iniciada com o novo conjunto de parâmetros.

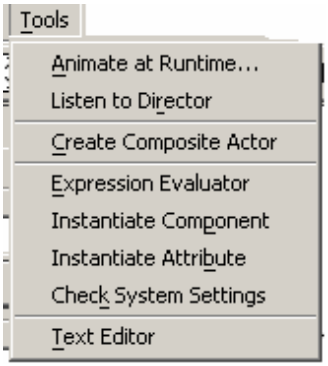

Figura 5.16. – Ferramentas do Vergil

O *menu de ferramentas do Vergil* permite que se anime a execução de um workflow (*Animate at Runtime*...), neste caso o usuário deverá indicar o intervalo de tempo de animação de cada ator. A animação propriamente dita consiste de um piscar intermitente do ator, indicando para o usuário que aquele ator sofrendo algum tipo de processamento na máquina de workflow Kepler. A janela *Create Composite Actor* é muito interessante, ela permite que o usuário defina não somente as portas de um ator

composto, como também seu comportamento que será definido através da composição de atores simples.

A janela *Expression Evaluator* opera de forma semelhante a um ambiente *shell*, ela oferece um ambiente textual que permite que o usuário construa e avalie qualquer tipo de expressão algébrica. Essa expressão poderá ser avaliada antes mesmo da execução do workflow, adicionalmente, ela poderá ser transferida para qualquer ator do tipo *Expression* de um workflow. A lista completa de operadores, funções (trigonométricas, matemáticas, array, matriz, registros, conversão de tipos, processamento de sinais, E/S, entre outras), variáveis pré-definidas e tipos de dados podem ser obtidos em Bhattacharyya et al. (2005a).

5.3.5.2 - Atores, Parâmetros e Variáveis

Atualmente existem aproximadamente 160 atores (simples), cada ator está associado com uma classe Java do arcabouço PtolemyII. Eles podem classificados como atores de controle de fluxo, entrada/saída de dados, filtro de dados, operações matemáticas, manipulação de imagens, entre outros. Determinados atores são capazes de encapsular aplicações externas, como por exemplo, serviços Web, funções de Grid, Matlab, consultas a bancos de dados, entre outros.

Os atores podem sem simples ou compostos. Os atores compostos são hierarquicamente aninhados formando um sub-workflow. Seu comportamento é muito parecido com o de um ator simples, ocorrendo apenas uma modificação no que tange a comunicação entre eles. A comunicação entre os atores simples, representada na figura 5.17, ocorre através de interfaces denominadas portas (*ports*). As portas podem ser classificadas como portas de entrada (*input ports*) ou portas de saída (*output ports*). Os atores, ou mais especificamente suas portas, são conectadas umas as outras por intermédio de canais (*channels*) pelos quais fluem as coleções de dados, uma coleção é composta por um ou vários tokens de dados.

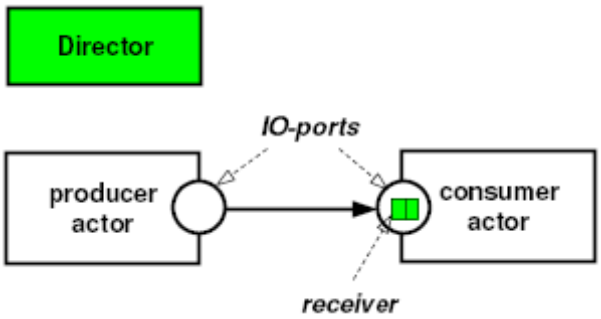

Figura 5.17. – exemplo de comunicação entre dois atores simples gerenciada por um diretor.

A comunicação entre os atores de um sub-workflow não é diretamente suportada pelo PtolemyII. Para contornar essa limitação os projetistas de um workflow científico devem utilizar ou dos seguintes artifícios. O primeiro é o mais simples e mais amplamente utilizado, ele consiste no uso de dois ou mais canais de comunicação. Um dos canais é utilizado para a troca de tokens de dados, enquanto que os outros canais são utilizados para o envio e recebimento de tokens especiais (de controle). Como principal desvantagem, destacamos a necessidade de definição explicitamente de todos os fluxos e das portas de comunicação.

A segunda abordagem, proposta por McPhillips e Bowers (2006a) é mais complexa e envolve o uso de um único canal onde os fluxos de controle e dados caminham juntos, entretanto é necessário delimitar o início e o fim de cada coleção de dados através do uso de tokens de controle. A figura 5.18 ilustra o funcionamento das coleções aninhadas de dados. A coleção *b* está aninhada na coleção *a*, elas fazem uso explícito de tokens de controle (*start* e *end*). As coleções delimitadas têm dois tipos de token, tokens de dados (d_i) e tokens de metadados (m_i). Essa abordagem permite que os tokens de metadados carreguem informações sobre todo o fluxo ou mesmo sobre a proveniência de dados, ela ainda permite que cada ator execute concorrentemente o conteúdo de uma coleção de dados. Por exemplo, cada ator é capaz de processar simultaneamente diferentes partes dos tokens de dados e metadados. Na figura 5.18, enquanto o ator 3 está processando o início da coleção a, o ator 2 está processando a coleção aninhada b e o ator 1 está processando o final da coleção a.

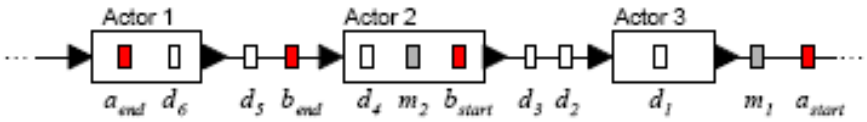

Figura 5.18. – Exemplo de fluxo misto no Kepler.

Os atores (simples ou compostos) possuem parâmetros (*parameter*), que são utilizados para configurar seu comportamento. Por exemplo, um ator do tipo filtro pode receber um conjunto de tokens através de uma determinada porta de entrada, em seguida aplicar o filtro definido pelo designer e finalmente, permitir a saída dos tokens que obedeçam à condição especificada no parâmetro.

5.3.5.3 - Diretores

Os diretores (conforme apresentado nos itens anteriores) são elementos chave, eles garantem não só a separação da semântica especificada no diagrama de blocos como também garantem como os atores irão se comunicar uns com os outros segundo as regras impostas pelo modelo de computação que ele representa. A separação é muito importante e vai ao encontro das características dos SGWf científicos estabelecidas por McPhillips e Bowers (2006a). Apesar de sua grande importância existem poucas referências sobre sua implementação.

5.3.5.4 - Proveniência e Gerência de dados no Kepler

Apesar do suporte à gerência e a proveniência de dados ser uma requisito importante para os SGWf científicos o Kepler ainda não oferece a essa característica de forma nativa. Para contornar essa limitação Altintas, Barney e Jaeger-Frank (2006) propõem um modelo genérico capaz de oferecê-la para diversos modelos de computação (Figura 5.13). Esse modelo será materializado sob a forma de um tipo especial de diretor, que também separará os contextos da modelagem e da execução.

Os componentes que oferecerão suporte à proveniência são o Provenance Framework e o Smart Re-run, eles foram desenvolvidos de forma a oferecer suporte aos projetos multidisciplinares do Kepler. O primeiro componente foi desenvolvido com o objetivo de registrar as informações relacionadas ao contexto de execução de um workflow, isto é, ele é capaz de registrar os dados de entrada e saída de um workflow, seus metadados,

os dados intermediários, além das definições do workflow e algumas informações relacionadas a sua execução. O segundo componente permite que uma determinada instância de workflow seja executada a partir de um determinado ponto. Por exemplo, Caso um usuário altere um parâmetro de um ator e execute o workflow novamente, não será necessária a re-execução dos atores inalterados. O Smart Re-run, leva em consideração os dados de proveniência armazenados e somente executará as partes do workflow que foram afetadas pela mudança de parâmetros. Altintas, Barney e Jaeger-Frank (2006) estão desenvolvendo um novo exemplo de wf de bioinformática que fará uso dos componentes propostos.

5.3.6 - Limitações do PtolomeyII-Kepler

O PtolemyII-Kepler é um ambiente muito amplo e complexo, ele permite que se construam workflows científicos para diversas áreas do conhecimento. Entretanto, apesar de apresentar inúmeras características singulares, ele também apresenta uma série de entraves que variam desde a quantidade (efetiva) de usuários até limitações da arquitetura, por exemplo, Ludäscher (2006) destaca a existência poucos desenvolvedores na equipe do Ptolemy, Gordenis et al(2005) enumera não mais que 30 usuários do Kepler e aproximadamente 10 workflows em produção. Nas próximas páginas apresentamos uma lista com as principais limitações, a saber:

1. Apesar da facilidade instalação, o arcabouço exige que o cientista tenha algum conhecimento sobre a configuração do ambiente Java no seu desktop caso este se dedique a alterar o código-fonte da máquina de execução ou desenvolver novos componentes. Além disso, o Kepler (versão avaliada) ainda é mono-usuário.
2. Apesar da simplicidade da interface Vergil, o design de workflows requer um algum esforço para identificar os modelos de computação mais adequados para cada tipo de problema, além disso, os cientistas precisam dominar uma série de conceitos ligados ao uso dos atores e ainda em relação à própria execução do workflow.
3. Atores, Diretores e demais componentes listados na lista de componentes (Kepler Ontology List) não apresentam qualquer tipo de ajuda, o help ainda reflete o mesmo conteúdo de versões anteriores. Dificultando ainda mais a construção de workflows. Apesar do mecanismo de busca de atores existentes, não há nenhuma meta-informação sobre os eles.
4. O PtolemyII somente suporta a execução de aplicações personalizadas, isto é, para que um programa seja executado pelo Kepler, ele precisa ser encapsulado como um ator. Esse é um fator limitante uma vez que nem sempre os cientistas dispõem de tempo ou dominam as técnicas de programação necessárias para fazê-lo.
5. Ausência de um repositório central de workflows. O Kepler somente é capaz de acessar e executar arquivos MoML locais, limitando a sua abrangência. A ausência do repositório não permite que vários cientistas acessem os workflows da organização simultaneamente, o compatilhamento de workflows ocorre através da simples troca de arquivos MoML.
6. Ausência de um repositório central de atores e diretores. Como não existe essa facilidade, os cientistas são responsáveis por manter os packages do arcabouço atualizados. Essa tarefa além de consumir tempo, é sensível a erros.

7. O PtolemyII não suporta monitoramento nativo, a tarefa é de responsabilidade do projetista. Assim, um novo grau de dificuldade é adicionado ao design dos workflows, pois além de gerenciar os fluxos de dados do experimento, o cientista terá que definir e gerenciar os fluxos de controle, além da geração de logs. Outra limitação, nem todos os diretores suportam o monitoramento. Na versão atual somente DT (Discrete Time) e PN (Process network) suportam-na.
8. A máquina de workflow não efetua nenhum tipo de validação ou verificação de erros do workflow antes de sua execução.
9. As notificações de erro são de difícil compreensão. As mensagens de erro não sofrem nenhum tipo de tratamento, o que dificulta a solução dos erros. A maioria das mensagens são exceptions geradas pelas classes Java, sendo apresentadas sob a forma de stack traces..
10. O Kepler não é escalável. Apesar de inúmeros esforços, a versão atual somente pode ser executada em uma máquina. A escalabilidade e o uso em ambiente de grid ainda estão sob estudos (Ludäscher, 2005, 2006).
11. Apesar da MoML ser XML, ela não é compatível com BPEL. Além disso, até o presente momento, o Kepler não apresentam nenhum mecanismo que busque essa integração.
12. Impossibilidade de comparar os workflows produzidos pelo Kepler com os padrões propostos por Aalst (2002, 2003).
13. A documentação do PtolemyII-Kepler é muito extensa e está ligeiramente ultrapassada, ela descreve os diretores e atores e demais componentes da versão 4.x (Bhattacharyya et al. 2005a, 2005b, 2005c).
14. Ausência de suporte nativo ao registro da proveniência de dados. A abordagem proposta por Altintas, Barney e Jaeger-Frank (2006) apesar de promissora, ainda está em fase de testes, os componentes que fazem o registro da proveniência foram implementados, porém até o momento nenhum workflow foi construído para validar o produto.
15. Ausência de suporte nativo para o uso de ontologias. Berkley et al (2005) propõem um extensão do Kepler para facilitar o uso de ontologias, a proposta têm como objetivos facilitar a autoria de workflows e o reuso por diferentes equipes de cientistas.

5.4 - OMII BPEL

O projeto OMII BPEL (Emmerich, 2005) é parte do programa OMII que pretende prover um software de Grid padrão e de software livre para aplicações científicas, este é uma proposta da Universidade College de Londres, coordenada pelo Departamento de Ciência da Computação desta instituição. Um dos principais objetivos desta ferramenta é aborada pelo OMII-BPEL que permite a combinação flexível de Serviços Web para formar workflows científicos. Como este tipo de workflow necessitam escala para suportar o volume de execuções a serem realizados dos workflows, além da freqüente alteração destes. Desta forma, uma solução que atenda a estas especificações necessita uma arquitetura com grande flexibilidade, sendo este o pricipal objetivo do OMII-BPEL.

O projeto desenvolveu um modelo para produção em ambientes científicos, monitorando e implantando um ambiente para a execução de workflows científicos

baseados em BPEL. O ambiente do OMII-BPEL incorpora uma versão do engine do ActiveBPEL (2008) que foi customizada e testado em diversos projetos de workflow científicos no Reino Unido (UK e-science projects).

Para a modelagem dos workflows a ferramenta acompanha um editor gráfico chamado Sedna. Este editor é baseado na plataforma Eclipse (2006). O editor incorpora um conjunto de extensões para a definição de workflows científicos, mas exporta o workflows na linguagem BPEL padrão, além de gerar os descritores necessários para a publicação do workflow no ActiveBPEL.

Na Figura 5.19. é apresentado o cliclo de vida de um workflow no OMII-BPEL. O workflow científico é criado e depurado no editor Sedna. Em seguida o workflow construído é convertido para BPEL e implantado no engine de execução (ActiveBPEL). O engine de execução interage com os serviços de monitoração e submissão de tarefas (jobs) do GridSAM que por sua vez submete a tarefa correspondente ao ambiente de Grid.

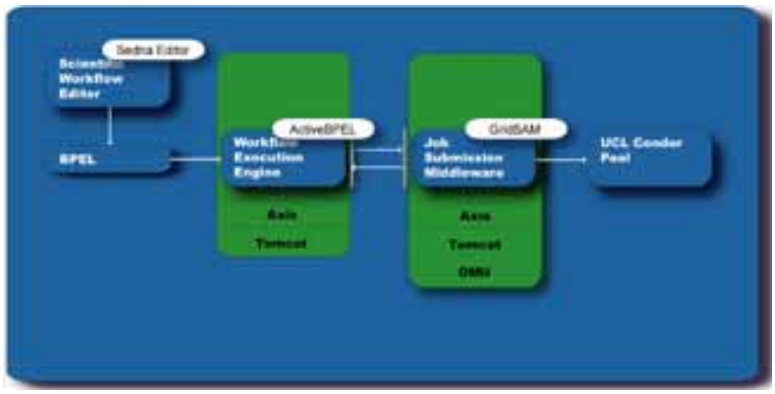

Figura 5.19. – Ciclo de vida de um workflow OMII

O OMII é um projeto recente que visa integrar o desenvolvimento de workflows científicos em BPEL com a robustes de um ambiente de Grid. A sua construção é calcada em soluções estabelecidas no mundo científico. Esta observação pode ser confirmada pelas escolhas do Eclipes, ActiveBPEL e o GridSAM. O uso do eclipse é o hoje o padrão da industria e da academia como framework para a construção de interfaces visuais de definição de programas. O ActiveBPEL é uma ferramenta de software livre para a execução de workflow BPEL. O GridSAM é um software livre permite o escalonamento e monitoração de tarefas na forma de serviços Web. Ele permite que diversos escalonadores de tarefas (DRM – Distributed Resource Managers) possam ser acessados como serviços Web, tais como Condor, Sun Grid Engine e o pbs.

6 - Exemplos de Workflows de Bioinformática

Dentre as inúmeras tarefas desempenhadas pelos bioinformatas destaca-se a concepção e a execução de workflows científicos. A concepção, conforme apresentado anteriormente está baseada na composição de programas próprios ou de terceiros. A composição não é uma tarefa simples, ela pode ser executada de duas formas principais. Na primeira, os cientistas recorrem a "pacotes" que possuem scripts que definem workflows. Também é relativamente comum o uso de linguagens de scripts, notadamente o PERL (Practical Extraction and Report Language) para implementar novos pipelines[6] ou mesmo desenvolver sistemas de análise e anotação de seqüências. Atualmente existem inúmeros sistemas de workflows que permitem a análise e anotação de seqüências, como por exemplo, o GARSA (Dávila et al., 2005), o STRINGRAY (Wagner et al., 2007), o SABIA (Almeida et. Al, 2004), GATO (Fujita et al. 2005), MidMon (Cruz et al, 2008a), OrthoSearch (Cruz et al, 2008b) Matriohska (Cruz et al, 2008c). entre outros. Também existem ferramentas que são exclusivamente utilizadas para a visualização de seqüências, por exemplo, Ártemis (Rutherford et al, 2000) e Apollo (Lewis et al, 2002). Nesta seção faremos uma breve descrição do GARSA e de seu workflow.

A segunda forma de composição utiliza sistemas de gerência de workflows científicos, e se diferencia da anterior graças ao uso de linguagens de definição de workflows e de máquinas de execução, entre outros. Atualmente, existem inúmeros sistemas de gerencia de workflow científicos Alguns operam em ambientes de grids de computadores, outros não. Ainda outros operam com serviços Web. Dentre eles apresentaremos, em detalhes, exemplos de workflows de bioinformática relacionados com os ambientes Kepler e ^{my}Grid.

6.1 - Promoter Indentification Workflow

Atualmente, diversos genomas total ou parcialmente completos estão disponíveis em bancos de dados de acesso público. Apesar do procedimento experimental de sequenciamento (de genes e genomas) ser rotineiro, não é possível dizer o mesmo com relação à anotação e interpretação dos dados experimentais. Neste contexto, inúmeras técnicas de extração do conhecimento podem ser aplicadas sobre as bases de dados, uma das técnicas que mais se destacam é a genômica comparativa (Hardison, 2003)

A genômica comparativa permite que se comparem diversos aspectos de uma seqüência. Por exemplo, similaridades, localização gênica, comprimento e número de regiões codificantes, quantidade e perfis de região não codificantes, entre outros. A comparação de seqüências é uma prática relativamente comum em bioinformática e neste sentido a presença dos workflows científicos se faz necessária.

O ambiente PtolemyII-Kepler possui dois exemplos de workflows de genômica comparativa: o IPT (Inferring Phylogenetic Trees) definido por McPhillips e Bowers (2005) e o PIW (Promoter Indentification Workflow) definido por Ludäscher, Altintas, 2006. Existem outros exemplos que pouco se relacionam com a bioinformática, eles estão voltadas para a Ecologia (um área da Biologia), portanto não apresentaremos

[6] O termo pipeline aparece muito frequentemente nos trabalhos de bioinformática, ele eventualmente é usado como sinônimo de workflow.

maiores detalhes de implementação uma vez que seus objetivos não se alinham com o foco desse relatório. Entretanto, apesar desse distanciamento, destacamos o potencial para futuros estudos desses exemplos. O primeiro exemplo é o BIC (Biodiversity Index Calculaletor) calcula os índices de biodiversidade através de funções e scripts que são executadas pela máquina de workflow. O segundo exemplo é o KNB (Knowledge Network Biodiversity) ainda está em fase de desenvolvimento, ele foi proposto por Berkley et. al (2005) e se diferencia de todos os outros, pois propõe uma extensão do Kepler através do uso de atores semânticos que em tempo de execução podem ser associados a ontologia de domínio.

Neste primeiro estudo de caso, apresentamos o experimento *in silico*, representado PIW, uma vez que é mais abrangente que os demais. O PIW faz parte do projeto SDM (Scientific Data Management Project), financiado pelo Departamento de Energia dos Estados Unidos (DOE) desde 2002. A versão atual foi desenvolvido no ambiente Kepler (Ludäscher, Altintas, 2003)[7] e, é capaz de executar seqüências de programas de bioinformática locais ou distribuídas (serviços Web) e exibir dados (textuais e gráficos) previamente transformados por atores especiais. Do ponto de vista biológico ele é voltado para o estudo da genômica comparativa e tem por objetivo facilitar a pesquisa de alguns biólogos que se dedicam ao estudo de como os organismos eucariotos respondem através de mudanças na expressão de certos genes quando expostos ao aumento da radiação.

Do ponto de vista conceitual, o trabalho executado pelos biólogos está dividido em oito etapas (Figura 6.1). Na primeira etapa, utilizam-se os resultados obtidos pela tecnologia de microarrays para encontrar o nível de expressão gênica de um grupo de genes. A segunda etapa consiste em obter os "accession numbers" dos genes que expressam alguma similaridade com os promotores previamente descritos na literatura. O agrupamento é produzido pela ferramenta Clusfavor.

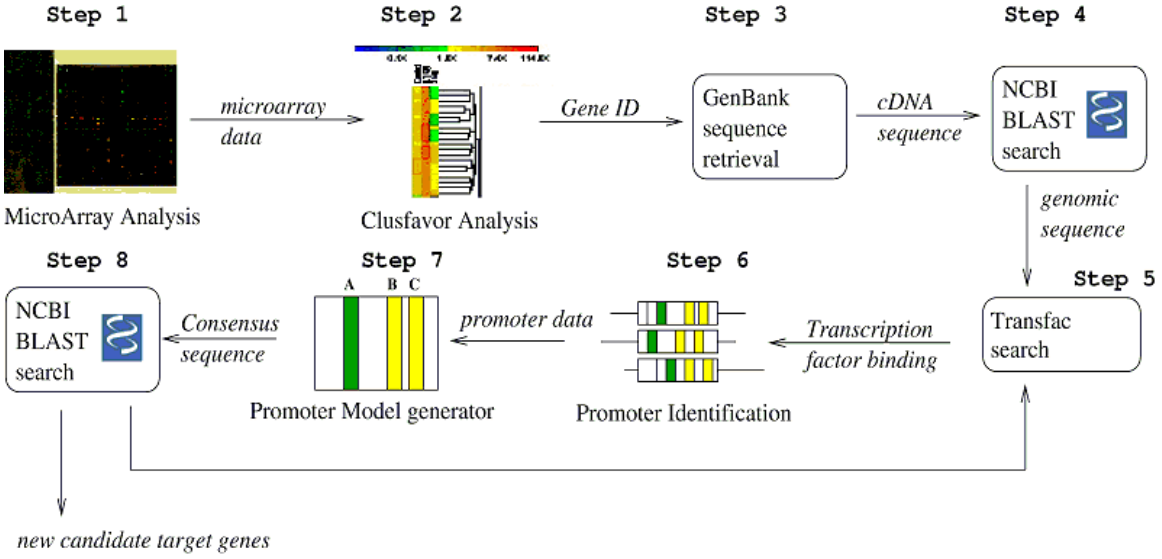

Figura 6.1 – Visão geral do experimento PIW

[7] Faz-se necessário salientar que existe uma versão antiga do PIW, ela é um pipeline (open-source), implementado sob a forma de CGIs, ele ainda pode ser acessada pela URL http://kbi.sdsc.edu/sciDAc-SDM/piw0.html

A terceira etapa consiste da recuperação de seqüências no GenBank e na posterior extração dos primeiros 300 pares de base de cada seqüência. Posteriormente, na quarta etapa, ocorrerá a análise de similaridade dos fragmentos de seqüências no BLAST do NCBI. Nesta etapa usam-se os serviços Web BLAST e FASTA oferecidos pelo DDBJ (DNA Data Bank of Japan) (DDBJ, 2006). Como resultados produzem-se seqüências homólogas alinhadas no formato FASTA, elas servem de input para uma nova busca em um banco de dados mais específico.

A quinta etapa tem como objetivo buscar os sítios dos fatores de transcrição (transcription factor binding sites) em um banco de dados de fatores regulatórios chamado Transfac (TRANSFAC, 2006). Entretanto, antes da busca propriamente dita, o workflow selecionará as duas melhores seqüências (best match), descartando as demais. Para cada uma dessas seqüências, a partir do ponto correlação (match) serão extraídos os 1500 pares de base anteriores e os 300 posteriores. Então, as duas seqüências ampliadas são submetidas à busca no Transfac. Os resultados das buscas são novamente submetidos a um segundo alinhamento no serviço Web BLAST.

A sexta e a sétima etapas são utilizadas na identificação dos sítios promotores e na construção de uma modelo de seqüência de consenso respectivamente. A oitava etapa executa o ClustalW, também como um serviço Web (DDBJ, 2006). O resultado desse processamento é o alinhamento múltiplo das seqüências produzidos inicialmente nas etapas 5 a 7. Finalmente, os resultados são enviados para componentes do Kepler que se encarregam de exibir, sob a forma de gráficos, as seqüências e seus respectivos sítios promotores, facilitando a análise que será realizada pelo biólogo.

Ao analisarmos o PIW e compararmos com a sua implementação no ambiente Kepler, percebemos que apesar da relativa facilidade de uso da ferramenta de edição de workflows (Vergil), a definição do workflow abstrato é complexa e exige refinados conhecimentos sobre o funcionamento dos atores, seu encadeamento e principalmente, sobre os formatos de dados produzidos pelos programas. Essa etapa é crucial, uma vez que são necessárias diversas transformações de dados.

O workflow PIW[8] está distribuído em vários níveis (sub-workflows que ficam aninhados no workflow de nível mais alto), essa metáfora visual é possível graças ao uso de atores compostos, oferecendo um mecanismo de escalabilidade no meio da complexidade do problema biológico. Adicionalmente, esse mecanismo também pode ser encarado como uma forma de encapsulamento, facilitando o reuso em novos workflows biológicos. Mesmo com o uso de atores compostos, a associação imediata entre as etapas da figura 6.1 e os atores do Kepler ainda é muito trabalhosa. Por exemplo, a figura 6.2 possui um alto nível de abstração, porém é possível reconhecer apenas dois atores simples, o primeiro (GeneAccessionNumberList) recebe a lista de accession numbers obtidos na etapa 2 da figura 6.2, o outro ator simples (Viz) é o encarregado da visualização gráficas dos alinhamentos executados pelo PIW. Os atores compostos (Setup, gene Sequence Processing, Run ClustalW, Merge and Discard, Log e Parse Clustal) representam, com algum grau de sobreposição, as etapas de 2 até 8 apresentadas na figura 6.2. É importante ressaltar que os atores compostos Setup e Log

[8] Nesta seção não apresentaremos todos os sub-workflows do PIW, pois muitos deles são muito parecidos uns com os outros, isto é, dizem respeito a transformações e formatações de dados. Portanto, dedicaremos especial atenção somente aos sub-workflows que julgamos mais significativos.

não fazem parte do experimento, nem estavam presentes na versões anteriores do Kepler.

A função do ator Log é aparentemente simples, ele se resume a registrar em arquivos texto do tipo ASCII os resultados de operações de alinhamento do BLAST, das transformações FASTA, das buscas no Transfac e as visulizações finais. Apesar da simplicidade, ele é responsável por registrar os dados produzidos pelos demais atores do workflow. Esse ator também executa uma série de transformações nos dados produzidos durante a execução do workflow, algumas das transformações envolvem a manipulação de strings, adição de timestamps, validações de arquivos e caminhos, entre outras. Observamos que esse ator também é responsável por registrar, parcialmente, a proveniência dos dados utilizados pelas diversas execuções do PIW. Cabe ressaltar que neste caso o registro da proveniência é deveras limitado, o ator atua como um auditor, isto é, ele não é capaz de armazena todas as informações de proveniência definidas na seção 2.5.

A função do ator Setup também é meramente administrativa, ela é uma exigência da tecnologia adotada pelo Kepler. Graças a este ator, é possível configurar a localização dos arquivos de log e dos arquivos de entrada de dados, obtidos a partir da análise dos microarrays.

No PIW, os atores são conectados através de portas de entrada e saída. Os fluxos de dados são representados sob a forma de linhas sólidas, podendo ser subdivididos quando estiverem conectados aos pequenos diamantes. Também é possível observar a presença de várias expressões, representadas sob a forma de pequenas caixas brancas. As expressões desempenham papel de grande importância no workflow, elas permitem que os formatos dos resultados de um serviço Web ou programa sejam transformados em formatos adequados para a entrada de dados de outros serviços ou programas. Essa característica, por si só já agrega grande dificuldade à construção do workflow abstrato, uma vez que o cientista não só deverá trabalhar por sobre os dados convenientes para o seu experimento, como também deverá reconhecer os formatos mais adequados para próximos programas do workflow.

As transformações de dados podem ser consideradas como o “calcanhar de Aquiles” de um workflow científico, são etapas complexas, demoradas e sensíveis a erros. As dificuldades em transformar os dados em conhecimento, têm um ponto de partida bem definido, elas iniciam-se na seleção das informações presentes nos bancos de dados públicos, entretanto não tem um final estabelecido. Os dados podem seguir caminhos dispares, como por exemplo, abastecer bancos de dados ditos “curados”; servir de dados intermediários para outros programas, entre outros. Essas características ressaltam a importância dos meta-descritores de dados, de esquemas de dados comuns, além descritores de linhagem dos dados.

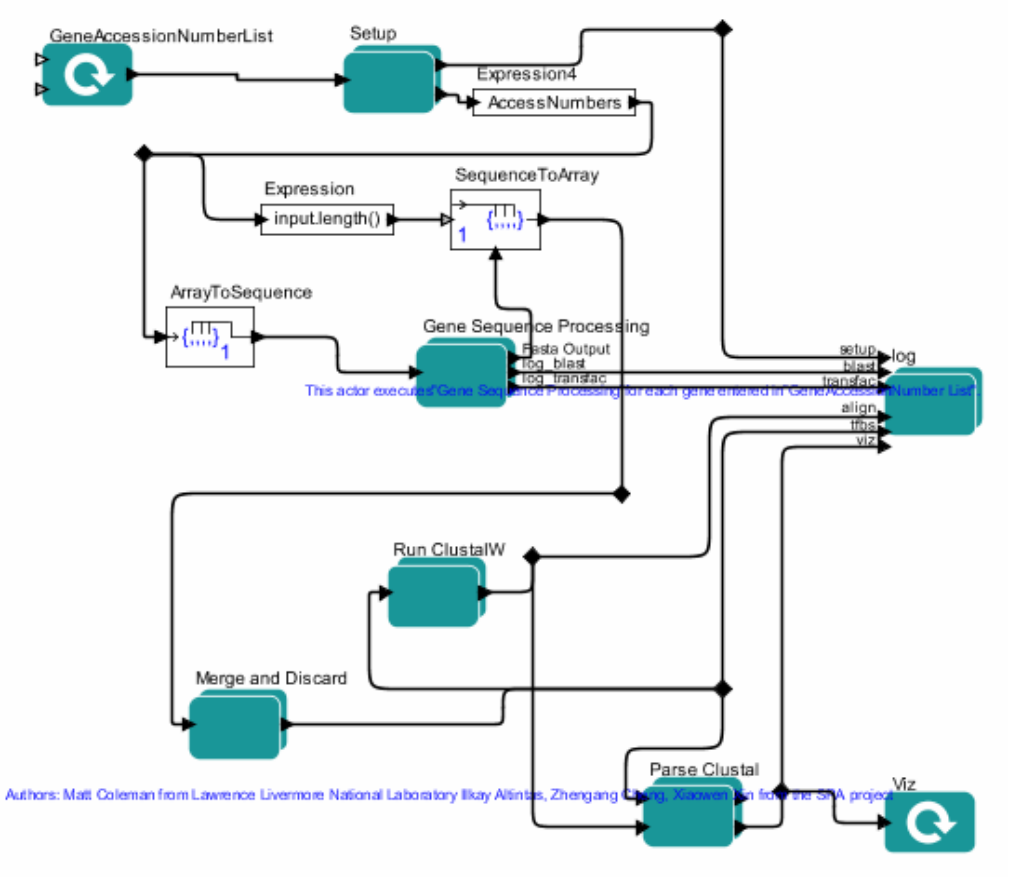

Figura 6.2 – Visão geral do Workflow PIW

Uma outra característica importante, previamente apresenta, diz respeito à aderência do PIW aos padrões de controle, comunicação e dados de workflow propostos por Aalst et al. (2002, 2003). Apesar ser possível perceber, ao menos neste nível de representação, a existência de padrões simples (*sequence*, *paralel split* e *simple merge*), não podemos afirmar categoricamente se a representação gráfica de fato indica um padrão de workflow. Este workflow, assim como os demais exemplos presentes no Kelpler, não são formalmente definidos segundo os moldes preconizados por Aaslt (*l.c.*). Na atual versão do Kepler (Alpha9), os workflows abstratos são armazenados como documentos XML, isto é, eles são representados segundo uma meta-linguagem de marcação proprietária (MoML). Até o momento, apesar de todos os nossos esforços, não foi possível localizar na literatura estudos que apresentam correlações entre a MoML e, por exemplo, redes de Petri, álgebra de processos, entre outras.

O ator composto Gene Sequence Processing é certamente o mais complexo de todos. Ele corresponde às etapas 3, 4, 5, 6 e parcialmente da sétima etapa do workflow PIW, ele está representado na Figura 6.2. Este ator, por sua vez, é composto por vários atores compostos (Debug, Get Sequence, Process Blast, Sychonyzer, Filter, Run Transfac) que possuem uma correlação mais ou menos direta com as etapas anteriormente citadas. Com relação às portas de entrada e saída não é necessário fazer maiores comentários, exceto pelo fato de que todas as saídas são utilizadas para abastecer os arquivos de log com informações (limitadas) de sobre a proveniência dos dados utilizados naquela execução do workflow.

De todos os atores presentes neste nível de abstração, os atores Process Blast e Synchronizer desempenham um papel central no processamento do workflow (Figura 6.3), eles são reesponsáveis pelas transformações de dados. O ator composto Process Blast está subdividido em 4 níveis (sub-workflows). Ele é capaz de fazer o parse dos resultados do BLAST, mas para que esse processamento alcance sucesso, é necessário converter as strings resultantes em documentos XML, a seguir, aplicam-se consultas XPath e XQuery para localizar as seqüências que contenham os sítios dos fatores de transcrição, uma vez localizadas, um processamento adicional é realizado, isto é, nesse ponto ocorre a inclusão das bases anteriores e posteriores ao ponto de correlação.

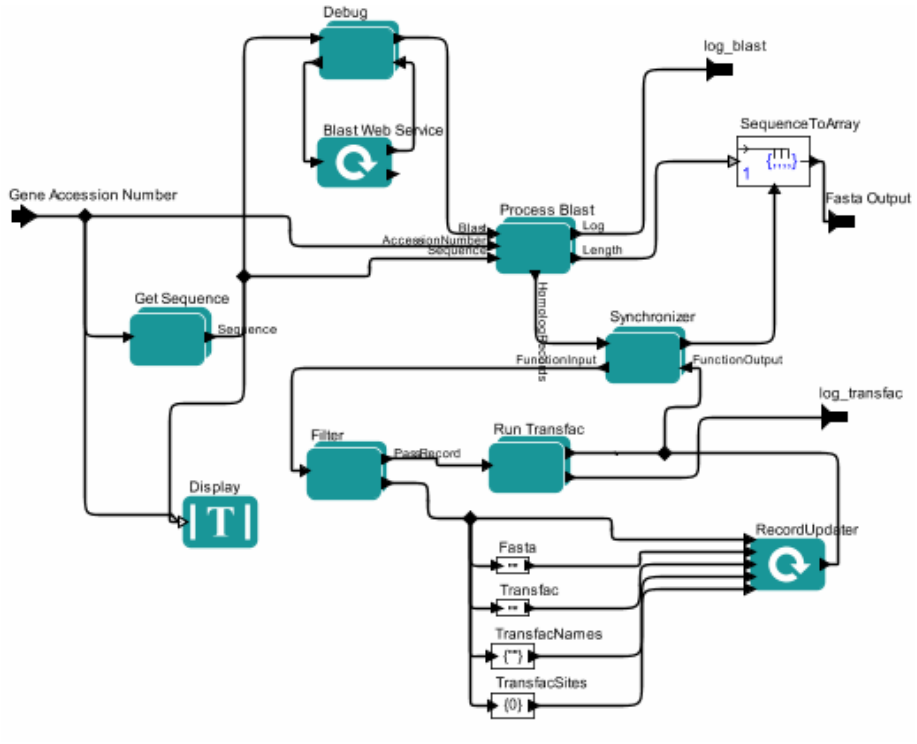

Figura 6.3 – Detalhes do ator composto Gene Sequence Processing

O ator composto Synchronizer (Figura 6.4) desempenha um papel admirável no PIW. Do ponto de vista sintático ele é bem simples, sendo composto por um pequeno conjunto de expressões e dois conjuntos de portas. Porém, do ponto de vista operacional ele possui as seguintes funções:

i - Sincronizar o funcionamento dos atores do workflow com o diretor (Process Network), esse diretor é usado para orquestrar a execução do workflow, os atores se comunicam através de mensagens que fluem em canais unidirecionais, elas são armazenadas em buffers locais e tratadas como uma fila do tipo FIFO, no caso um ator tentar ler uma fila vazia ele sofrerá um bloqueio até que haja algum input válido, na caso de escrita não ocorre bloqueio algum, pois as filas são consideradas infinitas;

ii - Sincronizar todos os fluxos de dados oriundos do serviço Web BLAST, essa função é muito importante uma vez que os serviços Web funcionam de modo assíncrono, além disso, podem ocorrer problemas de disponibilidade e conectividade;

iii - Empacotar todos os fluxos de seqüências homólogas produzidos pelo ator ProcessBlast, essa função utiliza as portas padrão (input e output), as demais funções fazem uso exclusivo das portas FunctionInput e FunctionOutput.

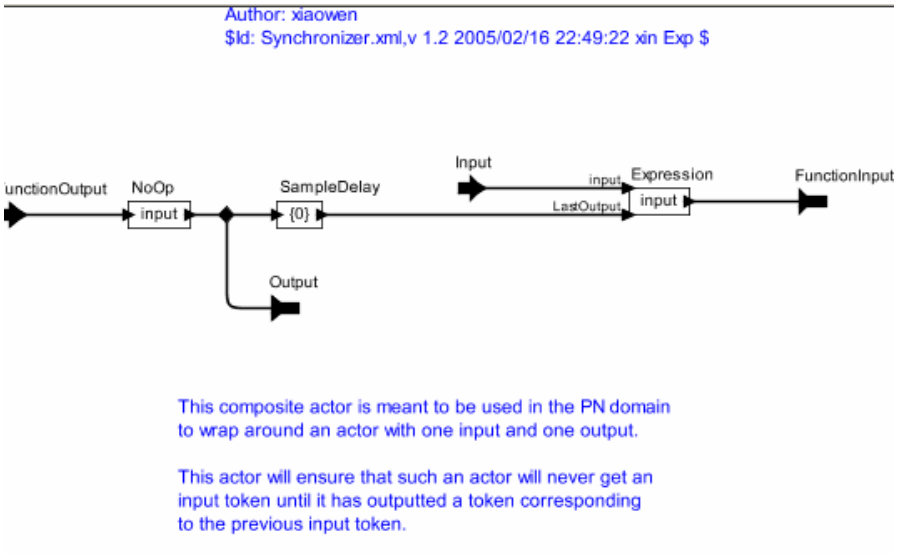

Figura 6.4 – Detalhes do ator composto Synchronizer

O ator RunClustalW, apesar de composto, é de fácil compreensão. Ele representa uma parte da oitava etapa do workflow PIW (a segunda parte dessa etapa é representada pelo ator Parse Clustal). Na figura 6.5 é possível verificar que o programa ClustalW está representado sob a forma de um serviço Web disponível no DDBJ. Porém, antes da sua invocação é necessário garantir que a formatação dos parâmetros de entrada estão compatíveis com a especificação do WSDL. Nesse caso, as expressões (1, 2 e 3) desempenham um papel primordial, elas recebem os dados de uma porta de entrada (*port1*) e fazem as transformações necessárias nos dados que serão enviados para o referido serviço. Na figura 6.5, também é possível observar que o resultado do alinhamento múltiplo realizado pelo ClustalW será enviado tanto para uma porta de saída (*port2*), quanto para um componente Display. É importante ressaltar que esse conteúdo poderá ser analisado pelo cientista, enquanto o workflow estiver em execução.

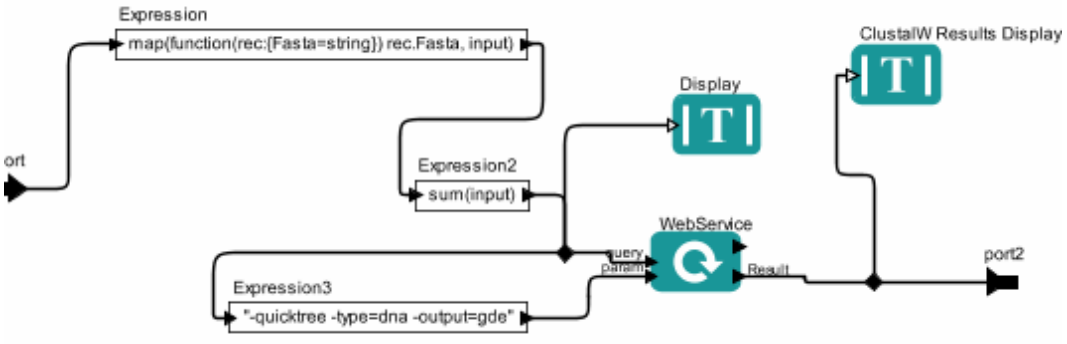

Figura 6.5 – Detalhes do ator composto ClustalW

Na próxima seção apresentaremos um segundo exemplo de workflow científico.

6.2 - GARSA

O GARSA (*Genomic Analyses Resources for Sequence Annotation*) é um ambiente com uma interface web desenvolvido com o objetivo de facilitar a análise, integração e apresentação das informações genômicas, através da concatenação de diversas ferramentas da bioinformática e bases de dados de seqüência. Os principais pontos que motivaram a criação do GARSA foram (Dávila et al., 2005):

i. Necessidade de interfaces integradas e amigáveis para realizar a anotação genômica;
ii. Necessidade de workflows para anotação genômica para web;
iii. Necessidade de um sistema de anotação genômica capaz de trabalhar com dados nos formatos GSS, EST e ORESTES;
iv. Necessidade de um sistema colaborativo com interface web que oferecesse recursos para autenticação de usuários.

O GARSA foi projetado especificamente para facilitar a análise de dados apresentando um workflow composto por pacotes de softwares para bioinformática selecionados e uma interface web intuitiva para os biólogos.

A plataforma deste sistema faz parte do consóricio BioWebDB (BioWebDB, 2008) e inclui Perl, Bioperl, CGI, Apache e MySQL, além de vários pacotes para bioinformática específicos para Linux.

O GARSA não é um sistema de gerência de workflows, pois não permite a instanciação de qualquer workflow científico. Ele se concentra num problema específico da bioinformática e possibilita a execução de um workflow específico, exibido na figura 6.6. Este workflow permite automatizar o processo de anotação genômica.

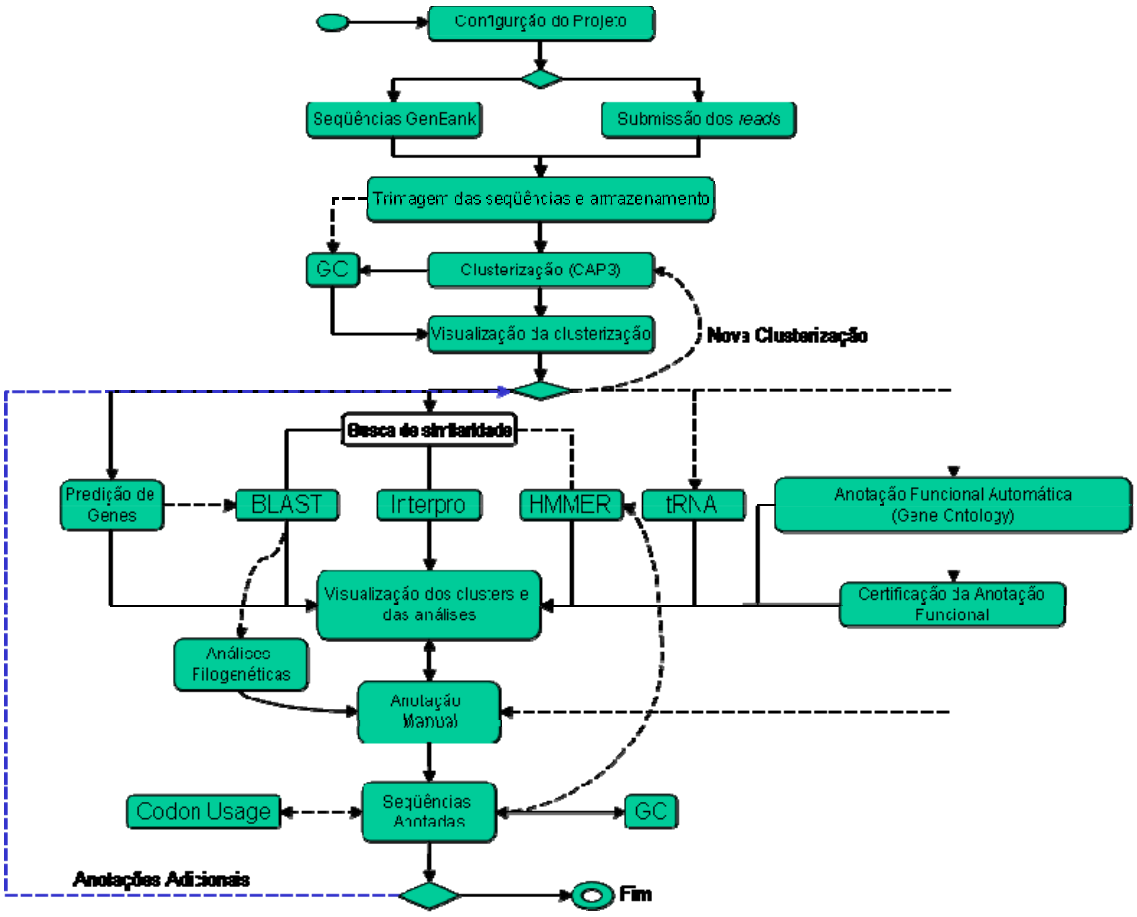

Figura 6.6 – Workflow do GARSA (BioWebDB, 2008)

Uma das primeiras etapas desse workflow é a análise dos dados de entrada que podem ser diretamente fornecidos pelo usuário ou capturados do GenBank Estes dados podem estar nos formatos EST, Orestes e GSS, representando seqüências de cromatogramas, seqüências do GenBank, arquivos Fasta armazenados localmente ou a combinação de todos estes.

Nos passos que se seguem, ocorrem diversos processamentos sobre estes dados. Para executar estes passos, o usuário deve configurar as ferramentas de sua preferência de acordo com cada etapa do workflow. Além disso, o usuário também deve realizar a configuração dos parâmetros necessários a cada processamento, conforme a ferramenta escolhida. A customização e execução de cada passo são conduzidas intuitivamente por meio de páginas exibidas seqüencialmente ao término de cada fase. Ao final da execução do projeto, são fornecidos os resultados que podem ser visualizados de diferentes formas.

Recentemente uma nova versão do GARSA, denominada STINGRAY (Wagner et al, 2007) vem sendo desenvolvida no consórcio BioWebDB, essa versão mantém as principais características do GARSA, incorporando algumas novidades como por exemplo o banco de dados genômicos GUS. Outra iniciativa apresentada por Cruz et al (2007) implementou parte do GARSA no Sistema de Workflow Kepler, dando indícios da viabilidade da plataforma para experimentos de Bioinfomática para o consórcio.

Software	Função	Versão
Phred	Trace quality	0.020425.c
Cross match	Remoção de seqüências de vetores	0.020425.c
CAP3	Clustering	3.0
Glimmer3	Gene prediction	3.0
YACOP	Gene prediction	2.0
Critica	Gene prediction	1.05
RBS Finder	Ribosomal binding site	1.0
Zcurve	Gene prediction	1.02
BLAST	Similarity	2.1.12
RpsBlast	Conserved Domain Similarity	2.1.12
Wu-Blast	Similarity	2.0
Interpro	Protein and domain similarity	3.3
HMMER	Disntant homology detection	2.3.2
Geecee	G+C content	2.9.0.6
Cusp	*Codon Usage*	2.9.0.6
tRNA-Scan	tRNA finder	1.23
Clustalw	Multiple alignment	1.83
Muscle	Multiple alignment	3.52
Probcons	Multiple alignment	1.10
WebLogo	Logo Alignment visualization	2.8
Phylip	Phylogeny	3.61

Tabela 6.1 – Softwares que podem ser utilizados no GARSA 1.8 (BioWebDB, 2006)

7 - Conclusão

Na apresentação deste relatório técnico foi possível observar que os worfklows científicos, voltados para a representação de experimentos biológicos *in silico*, possuem diversas questões em aberto. Como exposto, as principais questões envolvem: a gestão e armazenamento dos dados e workflows científico; a representação e descrição de dados e metadados envolvendo a definição de workflows que possam ser manipulados por diversos grupos de biólogos; a definição de um modelo formal para a construção dos workflows científicos; uma plataforma de desenho e execução destes workflows científicos; e por fim, uma linguagem padrão para a construção destes workflows.

Na sua estruturação o relatório distingue três abordagens para as questões relacionadas aos workflows científicos, são elas: a natureza do problema, o ferramental teórico para a formalização dos workflows e os principais engines para a execução dos workflows científicos. Desta forma, na sua seção 2, apresentamos os conceitos relacionados com os workflows científicos. Na seção 3, fazemos uma comparação entre os workflows comerciais e científicos para caracterizar suas principais diferenças. Na seção 4 os apresentamos o formalismo utilizado na representação de workflows e discutimos suas principais características. Na seção 5 reside a grande contribuição deste trabalho, onde os principais engines de execução de workflows da linguagem BPEL são apresentados; os dois dos principais projetos de engine de execução de workflows científicos (Kepler e ^{my}Grid) são detalhados e analisados no ciclo de vida de um workflow. Além disso, nesta seção, o protótipo OMII é descrito onde tenta abordar as mais novas questões que envolvem a construção de um engine de workflow científico. Por fim, na seção 6 apresentamos exemplos reais de workflows científicos.

Bibliografia

Aalst W. M. P. The application of petri nets to workflow management. The Journal of Circuits, Systems and Computers, 8(1):21–66, 1998.

Aalst W. M. P. Making work flow: On the application of petri nets to business process management. In J. Esparza and C. Lakos, (eds.) 23rd International Conference on Applications and Theory of Petri Nets, vol. 2360 of Lecture Notes in Computer Science, pp. 1–22. Springer-Verlag, Berlin, 2002

Aalst. W. van der, Hee, K. M. Van "Workflow Management: Models, Methods, and Systems" MIT Press. 2002.

Aalst, W. M. P., Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P. "Workflow Patterns". 2003.

Aalst. W. van der, Dumas, M., Hofstede, A. H. M. ter, "Web Service Composition Languages: Old Wine in New Bottles?" 2004.

Aalst. W. van der; Hofstede, A. H. M. ter, YAWL: Yet another workflow language. Information Systems, 30(4):245–275, 2005.

ActiveBPEL Engine Architecture, in: http://www.active-endpoints.com/open-source-architecture.htm, 2008.

Addis, M., Ferris, J., Greenwood, M., Li, P., Marvin, D., Oinn, T. and Wipat, A., "Experiences with e-Science workflow specification and enactment in bioinformatics", Proceedings of e-Science All Hands Meeting 2003, p. 459-466, East Midlands Conference Centre, Nottingham, 2003.

AGILENT. Agilebt Technologies. http://www.home.agilent.com/agilent/home.jspx?cc=US&lc=eng. 2008.

ALCHEMI, Alchemi .NET Grid computing Framework. Disponível em: http://www.alchemi.net/ , 2008

Almeida, L. G. Paixao, R. Souza, R. C., Barrieentos, F. J., Santos, M. T., et al."A System for Automated Bacterial (genome) Integrated Annotation- SABIA". Bioinformatics, 20: 2832-2833, 2004.

Alonso, G., Reinwald, B., Mohan, C., "Distributed Data Management inWorkflow Environments", Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE'97), 1997.

Altintas, I. et al. "A Modeling and Execution Environment for Distributed Scientific Workflows" 15th International Conference on Scientific and Statistical Database Management, p247-251, Cambridge. MA, 2003.

Altintas, I., Barney, O., & Jaeger-Frank, E. (2006). Provenance collection support in the Kepler Scientific Workflow System In International Provenance and Annotation Workshop (IPAW), LNCS, Provenance and Annotation of Data, 4145: 118-132, 2006.

Altintas, I., Barney, O., Jaeger-Frank, E. "Provenance Collection Support in the Kepler Scientific Workflow System", 2006.

Araujo R. M., Borges, M. R. da S. "Sistemas de Workflow" XX Jornada de Atualização de Informática. Congresso da SBC. Fortaleza, CE, Brasil. 2001.

Augustin, I., Lima, J. C. D., Yamin, A. C., "Computação pervasive: como programar aplicações", Tutorial do X Simpósio Brasileiro de Linguagens de Programação, pp 11-31, 2006.

Banerjee, S., "A Database Platform for Bioinformatics", 26th International Conference on Very Large Data Bases, p. 705 – 710, 2000.

Berkley, C., Bowers, S., Jones, M. B., Ludäscher, B., Schildhauer, M., Tao, J., "Incorporating Semantics in Scientific Workflow Authoring", 2005.

Bhattacharyya, S. S., Brooks, C., Cheong, E., Davis, J., Goel, M., Kienhuis, B., Lee, E. A., Liu, J., Liu, X., Muliadi, L., Neuendorffer, S., Reekie, J.; Smyth, N., Tsay, J., Vogel, B., Williams. W., Xiong, Y., Zhao, Y., Zheng. H., "Ptolemy II Heterogeneous Concurrent Modeling and Design in Java - Volume 1: Introduction to Ptolemy II". Document Version 5.0 for use with Ptolemy II 5.0, 2005a.

Baeten, J.C.M. A Brief History of Process Algebra. Report CSR 04-02, Eindhoven University of Technology, 2004.

Bhattacharyya, S. S., Brooks, C., Cheong, E., Davis, J., Goel, M., Kienhuis, B., Lee, E. A., Liu, J., Liu, X., Muliadi, L., Neuendorffer, S., Reekie, J.; Smyth, N., Tsay, J., Vogel, B., Williams. W., Xiong, Y., Zhao, Y., Zheng. H., "Ptolemy II Heterogeneous Concurrent Modeling and Design in Java - Volume 2: Ptolemy II Software Architecture". Document Version 5.0 for use with Ptolemy II 5.0, 2005b.

Bhattacharyya, S. S., Brooks, C., Cheong, E., Davis, J., Goel, M., Kienhuis, B., Lee, E. A., Liu, J., Liu, X., Muliadi, L., Neuendorffer, S., Reekie, J.; Smyth, N., Tsay, J., Vogel, B., Williams. W., Xiong, Y., Zhao, Y., Zheng. H., "Ptolemy II Heterogeneous Concurrent Modeling and Design in Java - Volume 3: Ptolemy II Domains". Document Version 5.0 for use with Ptolemy II 5.0, 2005c.

BioWebDB Portal 2006, in: http://www.biowebdb.org/index.html/, 2008.

BizTalk Server 2006, in:
http://www.microsoft.com/biztalk/techinfo/whitepapers/understanding.mspx, 2008.

Braghetto, K. R., Broinizi, M. E. B., Ferreira, J. E., Pu, C., "Simplifying the representation and execution of Workflow Patterns through Navigation Plan Definition Language", 2005.

Bowers, S. McPhillips, T., Ludaescher, B. Cohen, S. Davidson. S.B. “A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows”. In International Provenance and Annotation Workshop (IPAW), LNCS, 2006a.

Bowers, S. Ludaescher, B. Ngu, A. H.H. Critchlow. T. “Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow”, In IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow), 2006b.

Bowers, S., Ludaescher, B “A Calculus for Propagating Semantic Annotations through Scientific Workflow Queries,” EDBT 2006 Workshops, Query Languages and Query Processing (QLQP), 2006.

Carrer, H. “FORESTs: Eucalytus Genome Sequencing Project Consortium”. Disponível em http://forests.esalq.usp.br. 2002.

Cheong, E., Lee, E. A., Zhao, Y., Brooks, C. “A graphical Development and Simulation Environment for TinyOS-based Wireless Sensor Network” In 3rd ACM Conference on Embedded Networked Sensor System (SenSys2005) San Diego – Nov 2-4. 2005.

Costa, L. C.; Giraldi, G. A.; Schulze, B. R. Projeto Sistemas de Multi-Agentes e Suas Aplicações (http://virtual01.lncc.br/~luiscesar/PA2.htm), 2005.

Cruz, T. “Workflow: A tecnologia que vai revolucionar processos”, 2 ed. Ed. Atlas, São Paulo, Brasil. 2000.

Cruz, S. M. S. ; Silva, F. N. ; Gadelha jr., L. M. R. ; Cavalcanti, M. C. ; Campos, M. L. M.,; Mattoso,M. A Lightweight Middleware Monitor for Distributed Scientific Workflows. In: 3rd International Workshop on Workflow Systems in e-Science, 2008, Lyon, França. 3rd International Workshop on Workflow Systems in e-Science, 2008. v. 1. 2008a..

Cruz. S. M. S., Batista, V., Dávila, A. M. R., Silva E. Tosta, F., Vilela, C. Campos, M. L. M., Cuadrat, R., Tschoeke, D., Mattoso, M. OrthoSearch: A Scientific Workflow Approach to Detect Distant Homologies on Protozoans. A ser apresentado no ACM-SAC-2008 Track – Bio. Fortaleza – Ceará, 2008b.

Cruz. S. M. S., Barros, P. M., Bish, P., Campos, M. L. M., Provenance Services for distributed environments. A ser apresentado no CCGRID 2008. Lyon – França, 2008c.

Cruz, S. M. S. Silva, F. J. C da, Dávila A. M. R. Campos, M. L. M., Mattoso, M. Experiencing GARSA as a scientific workflow on grids environment In: Brazilian Symposium on Bioinformatics, (BSB´07), Angra dos Reis – Rio de Janeiro. 2007

Curbera, F., Goland, Y., Andrews, T., “Business Process Execution Language for Web Services v1.1”. Microsoft, BEA, IBM, Upddate - Feb-2007. Disponível em: http://www.ibm.com/developerworks/library/ws-bpel/. 2007.

Davidson, S., Crabtree, J., Brunk, *et. al.*, "K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources", IBM Systems Journal, Vol. 40, N. 2, p. 512-531, 2001.

Dávila, A. M., Lorenzini, D. M., Mendes, P. N., Satake, T. S., Sousa, G. R., Campos L. M. et al. "GARSA: genomic analysis resources for sequence annotation". Bioinformatics. 21(23):4302-4303, 2005.

DÁVILA, Alberto ; MENDES, Pablo Nascimento ; WAGNER, Glauber ; Tschoeke, D. A. ; Cuadrat, R. R. ; LIBERMAN, F. ; Matos, Luciana ; SATAKE, Thiago ; OCAÑA, K A C S ; TRIANA, O. ; GRISARD, Edmundo ; CAVALCANTI, M. C. ; CAMPOS, Maria Luiza Machado ; MATTOSO, Marta . ProtozoaDB: dynamic visualization and exploration of protozoan genomes. Nucleic Acids Research, v. 1, p. 1-6, 2007.

Dávila, A. Notas de Aula de Introdução à Bioinformática, IM-NCE/Fiocruz, 2006.

DBM - Departmento de Bioquímica Médica, Universidade Federal do Rio de Janeiro. Disponível em: <http://www.bioqmed.ufrj.br>. Acesso em: 18 Fevereiro 2008.

DDBJ – DNA Data Bank of Japan – WSDL list. Disponível em: http://www.xml.nig.ac.jp/wsdl/index.jsp. 2008.

Emmerich, W.; Butchart, B.; Chen, L.; Wassermann,B.; and Price, S.; "Grid Service Orchestration using the Business Process Execution Lenguage (BPEL)" in: http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/JoGC/Workflow/bpel.pdf, 2006.

Entrez WebService for NCBI, in: http://fiehnlab.ucdavis.edu/staff/scholz/, 2008.

Eshuis, R. Semantics and Verification of UML Activity Diagrams for Workflow Modelling. Ph.D. thesis, University of Twente, 2002.

Eshuis, R.; Dehnert, J. Reactive petri nets for workflow modeling. In W. M. P. van der Aalst and E. Best, (eds.) Application and Theory of Petri Nets 2003, vol. 2679 of Lecture Notes in Computer Science, pp. 295–314. Springer-Verlag, Berlin, Berlin, 2003.

Hoare, C.A.R. An axiomatic basis for computer programming. Communications of the ACM, 12:576–580, 1969.

Foster, I., Kesselman, C. (editors), "The Grid: Blueprint for a future computing infraestructure", Morgan Kaufmann, USA, 1999.

Foster, I., Vöckler, J., Wilde, M. e Zhao, Y., "The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration", CIDR, 2003.

Freefluo - Sourceforge. Disponível em http://freefluo.sourceforge.net/, visitado em Fev 2008.

Fujita, A., Massier, K. B., Durhham, A. M., Ferreira, C. E., Sogayar, M. C. “The GATO gene annotation tool for research laboratories”, Braz. J. Med. Biol. Res. 38(11) 1571-1574, 2005.

Galhardas, H., Florescu, D., Shasha, D., Simon, E. e Saita, C., "Improving Data Cleaning Quality Using a Data Lineage Facility", DMDW, 2001.

Galperin, M. Y., “The Molecular Biology Database Collection: 2007 update” Nucleic Acids Research, Vol. 35, pp1-3. 2007.

Georgakopoulos, D., Hornick, M. F., Sheth, A. P., “An Overview of workflow Mannagement: From Process Modeling to Workflow Automation Infraestructure”, Distributed and Paralle Databases, v.3, n.2, pp119-153, 1995.

GLOBUS, The Globus Alliance. Disponível em http://www.globus.org. 2008.

GMOD - Generic Model Organism Database Modular Schema – CHADO. Disponível em http://www.gmod.org/wiki/index.php/Schema/. 2008

Goble, C., "Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics", Workshop on Data Derivation and Provenance, Chicago, 2002.

Goble, C., Wroe, C., Stevens, R., and the myGrid consortium, “The myGrid Project: Services, Architecture and Demonstrator”, Proceedings UK e-Science All Hands Meeting 2003 Editors - Simon J Cox, p. 595-603, 2003.

Goderis, A., Sattler, U., Lord, P., Goble, C., “Seven Bottlenecks to Workflow Reuse and Repurposing”. ISWC, LNCS 3729, pp. 323-337, 2005.

Goodman, N., Rozen S., Stein, L. D., Smith, A. “The LabBase System for Data Management in Large Scale Biology Research Laboratories”. *Bioinformatics*, v. 14, n. 7, pp. 562-574, 1998.

Goodman, N., Rozen S., Stein, L. D. “The LabFlow System for Workflow Management in Large Scale Biology Research Laboratories”. *Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology (ISBM)*, pp. 69-77, Montreal, Quebec, 1998.

Greenwood, M., Goble, C., Stevens, R., Zhao, J., Addis, M., Marvin, D., Moreau, L. e Oinn, T., "Provenance of e-Science Experiments - experience from Bioinformatics", Proceedings of the UK OST e-Science 2nd AHM, 2003.

Hardison, R. C., “Comparative Genomics”. PLoS Biol. 1(2):e58. 2003.

Hollingsworth, D. - WFMC, “Workflow Management Coalition Terminalogy & Glossary” WFMC-TC 1011, 1995. Acesso em 01 de abr. de 2006. Disponível em.< http://www.wfmc.org/standards/docs/Ref_Model_10_years_on_Hollingsworth.pdf>

Jagadish, H. e Olken, F., "Database Management for Life Sciences Research", SIGMOD Record, vol. 33, 2004.

JBoss jBPM, in**:** http://www.jboss.com/products/jbpm, 2008.

Laud, A., Bhowmick, S. S., Cruz, P., Singh, D. T., Ramesh, G., "The gRNA: A Highly Programmable Infrastructure for Prototyping, Developing and Deploying Genomics-Centric Applications", Proceedings of 28th International Conference on Very Large Data Bases, p. 402-409, Hong Kong, China, (2002).

Lee, E. A., "What's Ahead for Embedded Software?" IEEE Computer, September 2000, pp. 18-26.

Lemos, M. "Workflow para Bioinformática". Tese de Doutorado – PUC-Rio, Rio de Janeiro, Brasil, 2004.

Lewis, S. E., Searle. S. M. J., Harris, N., Gibson, M., Iyer V. Richter J., et al. "Apollo: a sequence annotation editor". Genome Biology 3(12): 0082.1-0082.14. 2002.

Ludäscher, B. Altintas, I., Berkley, C., Higgins, D. , Jaeger-Frank, E. Jones, M. Lee, E. Tao, J. Zhao, Y., "Scientific Workflow Management and the Kepler System"Concurrency and Computation: Practice & Experience, 18(10), pp. 1039-1065, 2006.

Ludäscher, B., Altintas I. "On Providing Declarative Design and Programming Constructs for Scientific Workflows based on Process Networks" August. 2003.

McCarthy, J. A basis for a mathematical theory of computation. In P. Braffort and D. Hirshberg, editors, Computer Programming and Formal Systems, pages 33–70. North-Holland, Amsterdam, 1963.

McPhillips, T. M., Bowers, S. "An Approach for Pipelining Nested Collections in Scientific Workflows" SIGMOD Record, Vol. 34. No. 3. Sept. 2005.

McPhillips, T., Bowers, S., Ludäscher. B. "Collection-Oriented Scientific Workflows for Integrating and Analyzing Biological Data,"In 3rd International Conference on Data Integration for the Life Sciences (DILS), LNCS/LNBI 2006.

Marshack, R. T., "Workflow: applying automation to group processes" Coleman, D. Khanna, R., (eds), Groupware Technology and Application, 1 ed. Upper Saddle River, NJ, USA, Prentice Hall, 1995.

Moreau, L., Lord, P., Wroe, C., Stevens, R., Goble, C., Miles, S., Decker, K., Payne, T. e Papay, J., "Semantic and personalised service discovery", Procedings of Workshop on Knowledge Grid and Grid Intelligence (KGGI'03), em conjunto com IEEE/WIC International Conference on Web Intelligence/Intelligent Agent Technology, Halifax, Canada, 2003.

Moro, M., “Workflow”. Disponível em http://www.inf.ufrgs.br/~mirella/workflow/work.html e acessado em Fev. 2008.

myGrid. Disponível em http://www.mygrid.org.uk, visitado em 2008.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P., “Taverna: a tool for the composition and enactment of bioinformatics workflows”, Bioinformatics Journal, 20(17), p. 3045-3054, 2004.

Oracle BPEL Process Manager, in: http://www.oracle.com/technology/products/ias/bpel/index.html, 2008.

Petri, C. A. Kommunikation mit Automaten. Ph.D. thesis, Institut f˙ur Instrumentelle Mathematik, Bonn, 1962.

Plesums, C. “Introduction to Workflow” In Fischer, Liana, The Workflow Handbook, Future Strategies, 2002.

Rana, O. F., “Creating and Managing Distributed Scientific Workflows: Techniques and Tools”, 2005. Disponível em:

Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar N.. Workflow Control-Flow Patterns : A Revised View. BPM Center Report BPM-06-22, 2006.

Rutherford, K., Parkhill, J., Crook, J., Horsnell, T. Rice. P., Rajandream, M. A., Barrel, B., “Artemis: sequence visualization and annotation”. Bioinformatics; 16:944-945, 2000.

Sandeep, C., “Service Based Support for Scientific Workflows”. Dissertação de Mestrado- North Carolina State University, EUA, 2002.

Sangeeta, B., “An Evaluation of End-User Interfaces of Scientific Workflows Management Systems”, Dissertação de Mestrado- North Carolina State University, EUA, 2005.

Scott, D.S.; C. Strachey. Towards a mathematical semantics for computer languages. In J. Fox, editor, Proceedings Symposium Computers and Automata, pages 19–46. Polytechnic Institute of Brooklyn Press, 1971.

Santos, R. T., “O Ambiente 10+C Para A Definição E Execução De Workflows In Silico Através De Serviços Web”, Dissertação de Mestrado – COPPE/UFRJ, Rio de Janeiro, Brasil, 2004.

Sharman, N., Alpdemir, N., Ferris, J., Greenwood, M., Li, P. e Wroe, C., "The myGrid Information Model", Proceedings of UK e-science All Hands Meeting, 2004.

Silva, Fabricio Nogueira da; Cavalcanti, M. C.. “Intermediate Data Management for In-Silico Workflows Using Web Services”, Workshop de Teses e Dissertações em Banco de Dados, 2005, Uberlândia, MG, 2005.

Singh, M. P., Vouk, M. A., “Scientific Workflows: Scientific Computing Meets Transactional Workflows”, 1998.

Slominski, A., Gannon, D., Fox, G., “Introduction to Workflows and Use of Workflows in Grids and Grid Portals”, 2003. Disponível em:

Spooner, D. P., Cao, J., Jarvis, S. A., He, L., Nudd, G. R., “Perfomance-Aware workflow Management fro Grid Computing”. The Computer Journal, Vol. 48. n3. Oxford University Press. London. UK, 2005.

Sumpter, R., Whitepaper on Data Management, Lawrence Livermore National Laboratory. The IEEE Metadata Workshop, 1994.

WAGNER, Glauber ; Soriano, K. ; Jucá, H. ; BELLOZE, Kele Teixeira ; Tschoeke, D. A. ; Geronimo, G. A. ; Albrecht, F. ; MATTOSO, Marta Lima Queirós ; CAVALCANTI, M. C. ; CAMPOS, Maria Luiza Machado ; GRISARD, Edmundo ; Dávila, A. M. R. . STINGRAY: System for Integrated Genomic Resources and Analysis. In: International Workshop on Genomic Databases (IWGD'07), 2007, Angra dos Reis - RJ.

WARIA - Workflow And Reengineering International Association “WORKFLOW COMPARATIVE STUDY”. Volume II - Definition of Workflow Comparison Criteria. Disponível em: http://www.waria.com/books/study-volume1-2.htm#Volume_II__Definition_of_Workflow_Compar.

WebSphere Process Manager, in**:** http://www-306.ibm.com/software/integration/wps/features/, 2008.

Weske, M. Vossen, G., Medeiros, C. M. B. “Scientific Workflows Management: WASA Architecture ans Applications”, 03/1996-I Fachberich Angewandte Mathematik und Informatik, 1996.

WfMC, WfMC – Workflow Management Coalition “The workflow referece model”. Disponível em http://www.wfmc.org. 2008.

Wikipedia BPEL in**:** http://en.wikipedia.org/wiki/BPEL, 2008.

Wroe, C., Goble, C., Goderis, A., Lord, P., Miles, S., Papay, J., Alper, P., Moreau, L. “Recycling workflows and services through discovery and reuse”. 2005.

Taverna. Disponível emhttp://taverna.sourceforge.net/. 2008

Taverna Provenance Plugin. Disponível em http://www.mygrid.org.uk/wiki/Mygrid/TavernaProvenancePluginOneZero. 2008.

TRANSFAC – Transfac. Disponível em http://www.gene-regulation.com/pub/databases/transfac/doc/toc.html. 2008.

Yu, J., Buyya, R. “A Taxonomy of Scientific Workflow Systems for Grid Computing”, SIGMOD Record, Vol. 34, No. 3, Sept, 2005.

Zhao, J., Goble, C., Stevens, R. e Bechhofer, S., “Semantically Linking and Brownsing Provenance Logs for E-Science”, ICSNW, 2004.