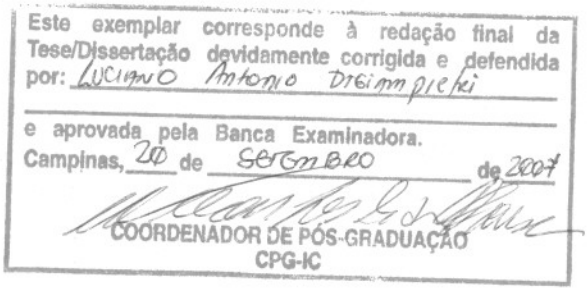
Gerenciamento de workflows científicos em bioinformática



Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Luciano Antonio Digiampietri e aprovada pela Banca Examinadora.

Campinas, 3 de agosto de 2007.

Prof. Dr. João Carlos Setubal (Orientador)

Profa. Dra. Claudia Bauzer Medeiros
(Co-orientadora)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues – CRB8a / 2116

Digiampietri, Luciano Antonio
D569g Gerenciamento de workflows científicos em bioinformática / Luciano Antonio Digiampietri -- Campinas, [S.P. :s.n.], 2007.

Orientador : João Carlos Setubal; Claudia Bauzer Medeiros
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Bioinformática. 2. Gerenciamento da informação. 3. Fluxo de trabalho. I. Setubal, João Carlos. II. Medeiros, Claudia Bauzer. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Management of bioinformatics scientific workflows

Palavras-chave em inglês (Keywords): 1. Bio-informatics. 2. Information management. 3. Workflow.

Área de concentração: Sistemas de Informação

Titulação: Doutor em Ciência da Computação

Banca examinadora: Prof. Dr. João Carlos Setubal (Virginia Tech)
Profa. Dra. Marta L. Queirós Mattoso (COPPE/UFRJ)
Prof. Dr. João Eduardo Ferreira (IME-USP)
Profa. Dra. Maria Beatriz Felgar de Toledo (IC-UNICAP)
Prof. Dr. Ricardo da Silva Torres (IC-UNICAMP)

Data da defesa: 03-08-2007

Programa de Pós-Graduação: Doutorado em Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 03 de agosto de 2007, pela Banca examinadora composta pelos Professores Doutores:

Profª. Drª. Marta Lima Queirós Mattoso
COPPE - UFRJ.

Prof. Dr. João Eduardo Ferreira
IME - USP.

Profª. Drª. Maria Beatriz Felgar de Toledo
IC - UNICAMP.

Prof. Dr. Ricardo da Silva Torres
IC - UNICAMP.

Prof. Dr. João Carlos Setubal
VBI – VIRGINIA TECH.

Instituto de Computação
Universidade Estadual de Campinas

Gerenciamento de workflows científicos em bioinformática

Luciano Antonio Digiampietri[1]

Agosto de 2007

Banca Examinadora:

- Prof. Dr. João Carlos Setubal (Orientador)
- Profa. Dra. Marta L. Queirós Mattoso
  Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ
- Prof. Dr. João Eduardo Ferreira
  Departamento de Ciência da Computação, IME – USP
- Profa. Dra. Maria Beatriz Felgar de Toledo
  Instituto de Computação – UNICAMP
- Prof. Dr. Ricardo da Silva Torres
  Instituto de Computação – UNICAMP
- Prof. Dr. Ivan Luiz Marques Ricarte (Suplente)
  Faculdade de Engenharia Elétrica e de Computação – UNICAMP
- Prof. Dr. Geovane Cayres Magalhães (Suplente)
  Instituto de Computação – UNICAMP

[1]Suporte financeiro de: Bolsa da CAPES de março de 2003 a abril de 2006 e Bolsa Microsoft Reserach de agosto de 2006 a julho de 2007

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Diz-me e eu esquecerei,
ensina-me e eu lembrar-me-ei,
envolve-me e eu aprenderei.

(provérbio chinês)

Resumo

Atividades em bioinformática estão crescendo por todo o mundo, acompanhadas por uma proliferação de dados e ferramentas. Isto traz novos desafios, por exemplo, como entender e organizar esses recursos, como compartilhar e re-usar experimentos bem sucedidos (ferramentas e dados), e como prover interoperabilidade entre dados e ferramentas de diferentes locais e utilizados por usuários com perfis distintos. Esta tese propõe uma infra-estrutura computacional para resolver tais problemas. A infra-estrutura permite projetar, re-usar, anotar, validar, compartilhar e documentar experimentos de bioinformática. Workflows científicos são os mecanismos utilizados para representar tais experimentos. Combinando pesquisa em bancos de dados, workflows científicos, inteligência artificial e Web semântica, a infra-estrutura se beneficia do uso de ontologias para permitir a especificação e anotação de workflows de bioinformática e para servir como base aos mecanismos de rastreabilidade. Além disso, ela usa técnicas de planejamento em inteligência artificial para prover as composições automática, iterativa e supervisionada de tarefas para satisfazer as necessidades dos diferentes tipos de usuários. Os aspectos de integração de dados e interoperabilidade são resolvidos combinando o uso de ontologias, mapeamento entre estruturas e algoritmos de casamento de interfaces. A infra-estrutura foi implementada em um protótipo e validada com dados reais de bioinformática.

Abstract

Bioinformatics activities are growing all over the world, following a proliferation of data and tools. This brings new challenges, such as how to understand and organize these resources, how to exchange and reuse successful experimental procedures (tools and data), and how to provide interoperability among data and tools across different sites, and used for users with distinct profiles. This thesis proposes a computational infrastructure to solve these problems. The infrastructure allows to design, reuse, annotate, validate, share and document bioinformatics experiments. Scientific workflows are the mechanisms used to represent these experiments. Combining research on databases, scientific workflows, artificial intelligence and semantic Web, the infrastructure takes advantage of ontologies to support the specification and annotation of bioinformatics workflows and, to serve as basis for traceability mechanisms. Moreover, it uses artificial intelligence planning techniques to support automatic, iterative and supervised composition of tasks to satisfy the needs of the different kinds of user. The data integration and interoperability aspects are solved combining the use of ontologies, structure mapping and interface matching algorithms. The infrastructure was implemented in a prototype and validated on real bioinformatics data.

Agradecimentos

Agradeço:

A **Deus** por me dar todas as oportunidades, amigos e desafios que me permitiram concluir esta tese.

A minha **família** que sempre me apoiou incondicionalmente. Em particular:

- A meus pais que sempre me transmitiram os valores do estudo e da educação e que sempre tiveram muita paciência comigo;
- A minha esposa que mesmo estando em outra cidade sempre se fez presente;
- A minha avó que com certeza não tem a mínima idéia do assunto da minha tese, mas sempre me apoiou integralmente.

A meus **orientadores** que estão nessa jornada comigo há muito tempo e que além da difícil (e paciente) tarefa de me orientarem, sempre valorizaram e incentivaram o meu trabalho. Aproveito para agradecer a cada um deles:

- João Carlos Setubal, meu orientador principal que me "nomeou" coordenador local do LBI durante sua ausência e me transmitiu diversos valores fundamentais para um pesquisador;
- Claudia Bauzer Medeiros, minha co-orientadora que me acolheu como orientando e se dedicou muito a minha pesquisa além de me incentivar a sempre enfrentar maiores desafios;
- Roger Barga, meu mentor na Microsoft Research que desde nosso primeiro contato foi muito prestativo e demonstrou muita confiança em minhas habilidades como pesquisador;
- Gonçalo Amarante Guimarães Pereira, meu orientador de iniciação científica que me apresentou a bioinformática e acreditou na minha capacidade de resolver desafios que nem eu mesmo acreditava que poderia.

Às agências de financiamento: **CAPES**, **FAPESP** e **CNPq** e à **Microsoft Research** que financiaram a pesquisa realizada.

Aos **revisores**, **editores** e **membros das bancas** dos eventos que participei, que, com suas visões imparciais, me ajudaram a manter um equilíbrio entre aspectos teóricos, práticos e minha paixão pela pesquisa.

Aos **amigos de república**, com os quais vivi muitas coisas nos últimos anos.

A meus (ex-)**professores**, especialmente àqueles que me envolveram com sua paixão pela ciência.

Aos **amigos do LIS** e demais **pesquisadores colaboradores** que estiveram sempre ao meu lado e me ajudaram a desenvolver minha pesquisa.

Aos **amigos do LBI** com os quais eu tive a honra de vasculhar o íntimo de diversos genomas.

Aos **amigos do LGE** que desde o meu segundo ano de graduação acreditaram na minha capacidade de desenvolver um grande projeto e contribuir com a ciência. Também por me ajudarem na especificação de exemplos e extração de requisitos para a minha tese.

Aos demais **amigos da UNICAMP** que desde a graduação participaram da minha jornada pela ciência.

Aos **demais amigos** que, mesmo sem entender direito o que é um doutorado (e muito menos o que é um workflow científico), me apoiaram muito e sempre me incentivaram a ir em frente.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

Workflows científicos [106] estão sendo cada vez mais adotados como meios para especificar e coordenar a execução de experimentos que envolvem participantes em locais distintos. Eles permitem a representação e execução de tarefas que usam dados e ferramentas heterogêneos [22].

Seu uso está se disseminando na área de e-Science, com aplicação em biodiversidade, física, química ou astronomia. A bioinformática, base dos estudos de caso da tese, foi um dos primeiros domínios científicos a adotar tais workflows [73, 99].

Eles são utilizados em experimentos *in silico* em diversos laboratórios, como nos Laboratórios de Genômica e Expressão (LGE) [62] e de Bioinformática (LBI) [61] da Universidade Estadual de Campinas. O LBI foi o primeiro laboratório brasileiro de bioinformática, sendo responsável pela coordenação da montagem e anotação do genoma da *Xylella fastidiosa* [92]. Este trabalho envolveu mais de 30 laboratórios do estado de São Paulo. Cada um desempenhou parte das tarefas, enquanto que a integração e validação dos resultados foram centrados no LBI. Subseqüentemente, o laboratório exerceu a mesma função em outros projetos análogos, desenvolvendo procedimentos de bioinformática envolvendo genomas ligados à agricultura (por exemplo, o da cana-de-açúcar). Esta experiência evoluiu naturalmente para soluções na especificação e implementação de uma infra-estrutura computacional para o gerenciamento de workflows científicos em bioinformática [32].

Workflows científicos diferem de workflows de negócio em diversos aspectos [106]. Particularmente em bioinformática, eles caracterizam-se pelo alto grau de intervenção humana na sua execução. Além disso, como a bioinformática ainda é uma área nova, ainda não há um consenso bem definido sobre como as tarefas devem ser executadas e como os resultados devem ser anotados [66]. Com isto, há uma multiplicidade de possibilidades de

workflows que podem ser projetados para desenvolver uma mesma função. Esta variedade de opções começa no nível mais abstrato de especificação (tipos de procedimentos a serem adotados) e prossegue até o nível de implementação (escolha de ferramentas para executar uma atividade, ou uso de conversores de dados adequados). Finalmente, vários workflows podem não terminar ou fornecer resultados não conclusivos. É preciso, assim, manter um registro não apenas das execuções bem sucedidas como também daquelas defeituosas.

O projeto de workflows científicos em bioinformática é tipicamente manual e realizado por meio de especificação e composição de atividades, que podem ser definidas de várias formas, por exemplo usando linguagens de *script* ou, mais recentemente, invocando serviços Web [22]. A ampla aceitação de tais serviços neste domínio depende da disposição da comunidade de biólogos em propor uma padronização para eles. Isto inclui a construção de ontologias, o desenvolvimento de repositórios de ferramentas específicos para biologia e a definição das interfaces para serviços comuns [46].

A composição manual é uma atividade árdua e suscetível a erros, mesmo quando usando serviços Web. Além do mais, em bioinformática, devido à constante evolução da área e a explosão combinatória de alternativas, há tantas possibilidades para a construção de workflows que é inviável computar e comparar todas elas. Assim, há uma crescente demanda por soluções que ajudem os cientistas a projetar os workflows desejados.

Outro aspecto importante está ligado à proveniência dos dados e ferramentas utilizados em cada experimento. Este tipo de informação é fundamental para que um experimento possa ser reproduzido, além de assegurar sua qualidade [17]. Além da necessidade de funcionalidades de proveniência, os sistemas de workflows devem prover mecanismos de rastreabilidade para que o usuário possa analisar detalhes dos experimentos já executados. Isto requer ferramentas para re-execução de partes dos experimentos, consultas sobre resultados parciais e parâmetros utilizados, etc.

Todos esses requisitos computacionais – para projeto, gerenciamento e rastreabilidade – apresentam desafios em computação.

Outra característica presente em laboratórios de bioinformática é a diversidade de usuários. Destacamos três tipos de usuários de acordo com suas necessidades e padrões de trabalho: o biólogo, o bioinformata e o cientista ou engenheiro da computação. O biólogo tipicamente não possui muitos conhecimentos em desenvolvimento de software e, desta forma, precisa de ferramentas que simplifiquem a especificação e execução de experimentos. O bioinformata possui conhecimentos em biologia e computação e assim está apto a montar workflows complexos que serão utilizados pelos biólogos. Já o cientista ou engenheiro da computação tem os conhecimentos para desenvolver todas as atividades básicas que poderão ser utilizadas pelos biólogos e bioinformatas. Esta visão, por mais simplificada que seja, ajuda a mostrar que os diferentes usuários presentes em laboratórios de bioinformática possuem requisitos muito distintos.

Há diversos sistemas para gerenciar experimentos de bioinformática (por exemplo, [29, 85, 96, 101]). Esses sistemas cumprem seus propósitos fornecendo ferramentas para a montagem e anotação de um único projeto genoma ou provendo modelos que criam uma visão integrada, porém simplificada, de diversos projetos genoma. Porém, ainda há aspectos de pesquisa em aberto no que tange o fornecimento de mecanismos para a integração de dados e a interoperabilidade de ferramentas sem restringir o modelo de dados que é utilizado em cada laboratório. Outro aspecto importante está relacionado à criação dos workflows [22, 40, 65, 99]. Há diversas propostas, variando do fornecimento de ferramentas que facilitam a busca durante a composição manual até sistemas que produzem workflows automaticamente. Exemplos de tópicos em aberto quanto à criação de workflows são: (i) como prover ferramentas adequadas a cada tipo de usuário quando consideramos um ambiente distribuído com perfis de usuários distintos; ou (ii) como utilizar o conhecimento do domínio da aplicação para melhorar a qualidade dos workflows gerados.

Outro tópico que apresenta aspectos de pesquisa em aberto é a proveniência e rastreabilidade dos dados e ferramentas de cada experimento. Ambientes laboratoriais de bioinformática são muito dinâmicos. A todo instante, novos dados e ferramentas são criados. Além disso, a anotação de genomas exige uma intensa comparação entre os dados novos e os dados já anotados. Desta forma, ferramentas de consulta que visem levantar dados de proveniência são fundamentais para a validação desses dados e para assegurar a sua qualidade. Para cada dado ou ferramenta, um usuário precisa conhecer *quando* o dado foi gerado, *quem* produziu esse dado, *onde* o dado foi gerado e *como* esse dado foi gerado. Para responder a essas perguntas um sistema deve, além de conter anotações sobre a proveniência dos dados, possuir o armazenamento detalhado da execução de cada workflow. Alguns tópicos de pesquisa em aberto nessa área são: especificar um modelo de armazenamento de experimentos que contenha todos esses tipos de informação e fornecer ferramentas que permitam a rastreabilidade de cada dado ou experimento.

Outro tópico relevante é a especificação formal de workflows. Existem duas propostas principais que discutem este tipo de especificação: a baseada em redes de Petri [1, 2] e a baseada em álgebra de processos [15]. A utilização de arcabouços formais para a especificação de workflows está fora do contexto desta tese.

1.2 Aspectos de Pesquisa Envolvidos

O principal desafio enfrentado por esta tese é facilitar a especificação, reuso, documentação, validação e compartilhamento de experimentos científicos de bioinformática. Partindo deste desafio, nossa pesquisa se concentrou em cinco aspectos: especificação de experimentos, anotação, integração de dados, interoperabilidade e rastreabilidade.

Especificação de experimentos. Nesta tese supomos que os experimentos científicos podem ser representados sob a forma de workflows científicos [106]. Esta hipótese está sendo usada por diversos grupos (por exemplo, [12, 49, 101]) para documentar ou executar esse tipo de experimento. O WFMC (Workflow Management Coalition) [51] definiu um modelo genérico de representação de workflows para prover interoperabilidade entre diferentes sistemas de workflows. No laboratório de Sistemas de Informação (LIS) [63] da UNICAMP foi desenvolvida uma extensão desse modelo, descrita em [71, 84], que permite a representação e o compartilhamento de workflows em vários níveis de abstração. Partindo deste modelo, esta tese enfrenta os seguintes desafios:

1. Como complementar o modelo existente [71, 84] de forma a agregar informações semânticas para facilitar o entendimento dos experimentos;

2. Como facilitar a integração de dados e interoperabilidade de ferramentas para a construção dos workflows supondo que os dados são armazenados em estruturas diferentes e as ferramentas não utilizam interfaces padronizadas;

3. Como adicionar dados de proveniência aos experimentos de forma a facilitar as consultas e possibilitar a rastreabilidade;

4. Como identificar as estratégias de composição de atividades que podem ser usadas para ajudar os diversos tipos de usuários a construírem seus workflows.

A especificação de um modelo de representação de workflows utilizou mecanismos da Web Semântica para enriquecer o conhecimento sobre os dados e as ferramentas. Já os mecanismos de composição de atividades foram desenvolvidos estendendo métodos de planejamento em Inteligência Artificial [65].

Anotação de dados e ferramentas. Um dos desafios na anotação de dados e ferramentas que são compartilhados entre diversos usuários é o fornecimento de um vocabulário adequado (e compartilhado) para a anotação dos mesmos. Este desafio é abordado na tese a partir da construção e uso de ontologias. Ontologias estão sendo amplamente utilizadas como mecanismos que fornecem um vocabulário uniforme de conceitos e o relacionamento entre esses conceitos. O trabalho exigiu a criação de ontologias (de domínio e de serviços) que: (i) forneçam um vocabulário adequado para a anotação dos dados e ferramentas; (ii) auxiliem usuários a construir workflows, reusando total ou parcialmente soluções desenvolvidas por outros.

Integração de dados. O grande crescimento dos dados sobre genomas (seqüências de DNA, genes, proteínas, etc) e das ferramentas disponíveis (muitas na forma de serviços Web) traz novos desafios: como aproveitar e integrar os dados disponíveis e prover interoperabilidade entre as ferramentas existentes. A anotação de dados seguindo conceitos

de ontologias consensuais permite a redução da ambigüidade e facilita seu entendimento. Porém, laboratórios distintos costumam disponibilizar seus dados de maneira heterogênea, o que dificulta o compartilhamento e o reuso. Esta tese enfrenta este desafio combinando técnicas de mapeamento de estruturas de dados [13] e o uso de ontologias.

Interoperabilidade. Nos últimos anos, diversas ferramentas para processamento de atividades em bioinformática foram disponibilizadas na Internet, sendo que muitas na forma de Serviços Web. A falta de padronização das interfaces dessas ferramentas e da descrição de suas funcionalidades dificulta seu uso. Para prover interoperabilidade entre as ferramentas, utilizamos uma estratégia composta de três abordagens. A primeira é a anotação de cada uma das operações de uma ferramenta (ou serviço) conforme nossa ontologia de serviços. A segunda abordagem é a anotação da interface de cada operação (parâmetros e tipos de resultados produzidos) seguindo nossa ontologia de domínio. Por fim, utilizamos algoritmos de casamento de interfaces [88] para verificar a compatibilidade semântica e sintática entre interfaces.

Rastreabilidade é a habilidade de se rastrear o processo (dados e ferramentas) no qual um objeto está envolvido. Mecanismos de rastreabilidade são comumente encontrados em, por exemplo, trabalhos de engenharia de software, comércio eletrônico ou cadeias produtivas. Em bioinformática há dois propósitos para se prover rastreabilidade: assegurar a qualidade de um experimento e permitir consultas mais elaboradas sobre todos os dados e ferramentas envolvidos na produção de um resultado. O desafio quanto a este tema é o desenvolvimento de mecanismos que permitam tais tipos de rastreabilidade. A solução adotada foi adequar o modelo de representação de workflows [71, 84] e proveniência [11] de forma a possibilitar a rastreabilidade e prover ferramentas que facilitem a navegação dentro de um experimento.

Com os resultados obtidos na pesquisa em cada um dos aspectos citados, especificamos e prototipamos uma infra-estrutura para o gerenciamento de experimentos científicos de bioinformática. Esta infra-estrutura ajuda a solucionar os problemas em aberto citados na motivação, estendendo as funcionalidades encontradas em outros sistemas com objetivos semelhantes.

1.3 Objetivos e Contribuições

O trabalho apresentado nesta tese resultou na definição da arquitetura apresentada na Figura 1.1. A arquitetura é composta por quatro camadas principais: repositórios; gerenciador de dados; módulos de processamento; e interface. Cada uma das publicações desta tese contribui com partes desta arquitetura. Esta seção utiliza a Figura 1.1 para apresentar três cenário de uso e situar as contribuições e as publicações no contexto geral da tese.

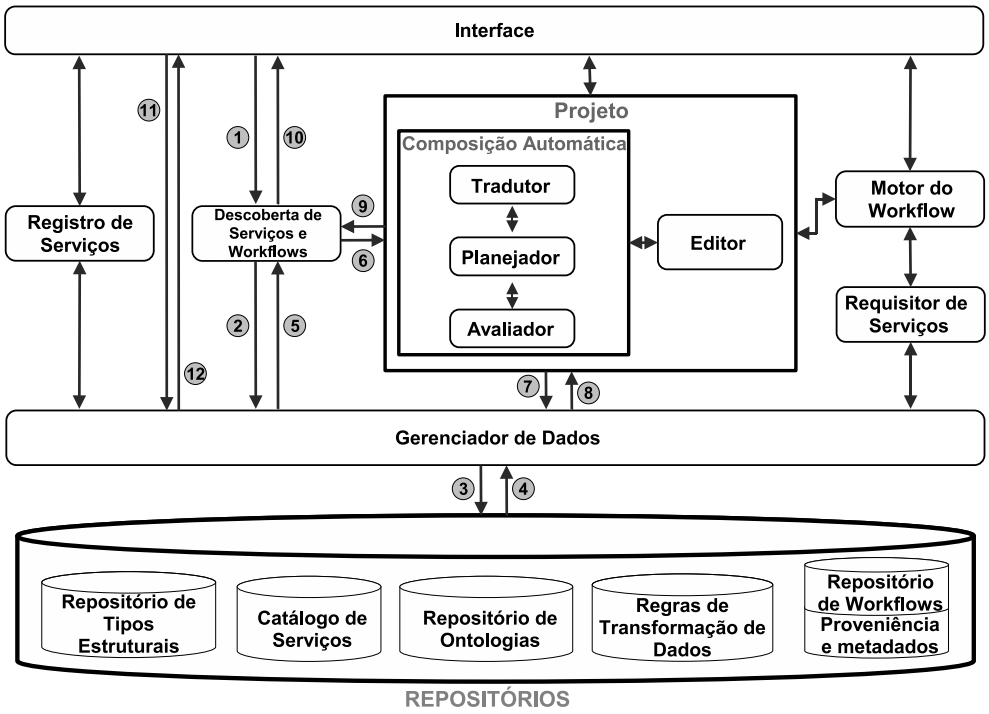


Figura 1.1: Arquitetura do sistema

Cenário de uso 1: usuário consulta o sistema para encontrar um serviço ou workflow desejado e existe um ou mais serviços ou workflows que satisfaçam os critérios de consulta do usuário. Toda interação entre o usuário e o sistema é feita via o módulo Interface. A Interface envia a consulta do usuário para o módulo de Descoberta de Serviços e Workflows (1). Este módulo encaminha a consulta para o Gerenciador de Dados (2) que consulta os repositórios de Ontologias, Workflows e o Catálogo de Serviços (3). Os serviços e workflows resultantes são encaminhados para o Gerenciador de Dados (4) que os envia para o módulo de Descoberta de Serviços e Workflows (5). Este módulo entrega esses resultados a Interface (10) que os apresenta ao usuário.

Cenário de uso 2: usuário consulta o sistema para encontrar um serviço ou workflow desejado e não existe um serviço ou workflow que satisfaça os critérios de consulta. Este cenário é muito parecido com o anterior, porém, ao não se encontrar um serviço ou workflow satisfatório, o módulo de Composição Automática será acionado para gerar, via planejamento, workflows que satisfaçam a necessidade do usuário. As mensagens de (1) até (5) neste cenário são iguais às do cenário 1, com a única diferença que as mensagens (4) e (5) dizem que não há serviço ou workflow satisfatório. Ao receber a mensagem (5) o módulo de Descoberta de Serviços e Workflows solicita ao módulo Composição Automática (6) a geração de novos workflows. Este módulo consulta o Gerenciador de Dados (7) para

obter os dados necessários ao planejamento. O Gerenciador de Dados encaminha esta consulta para os repositórios: Catálogo de Serviços, Repositório de Ontologias, Regras de Transformação de Dados e Repositório de Workflows (3). O Gerenciador de Dados recebe os resultados da consulta (4) e os envia ao módulo Composição Automática (8). Com estes dados o módulo de Composição Automática projeta novos workflows e os envia para o módulo de Descoberta de Serviços e Workflows (9) que por sua vez os entrega a Interface (10), onde são apresentados ao usuário.

Cenário de uso 3: usuário deseja consultar dados de proveniência de um certo experimento. Para consultar dados de proveniência ou anotar experimentos o usuário envia consultas, via Interface, para o Gerenciador de Dados (11) que traduz essas consultas, considerando aspectos de rastreabilidade, e consulta o Repositório de Workflows, Proveniência e Metadados (3). O Gerenciador de Dados recebe os resultados da consulta (4) e os encaminha para a Interface (12), onde esses dados são apresentados ao usuário.

O principal objetivo da tese é facilitar o trabalho de usuários em laboratórios de bioinformática (biólogos, bioinformatas e cientistas da computação) no que tange a construção, documentação, reuso e gerenciamento de experimentos, ferramentas e dados. Este objetivo deve ser atingido tanto em ambientes centralizados quanto distribuídos. As soluções adotadas para os diversos desafios apontados se baseiam em quatro eixos: (i) o uso de workflows científicos como a base para a especificação e execução de tarefas em um ambiente laboratorial distribuído; (ii) a adoção de ontologias consensuais, como forma de permitir compartilhamento de recursos, integração e interoperabilidade; (iii) o armazenamento, em bancos de dados, de ontologias e workflows, em diversos níveis de abstração (camada Repositórios da Figura 1.1), e (iv) o uso de planejamento em Inteligência Artificial (IA) para facilitar a construção automática de workflows (caixa rotulada Composição Automática, na Figura 1.1).

Cada atividade dentro de um workflow pode ser executada pela invocação de um serviço Web ou por outro (sub-)workflow. Assim, usamos os termos "composição de serviços" e "composição/construção de workflow" indistintamente.

Partimos ainda da hipótese que o problema de composição automática e semi-automática de workflows pode ser visto como um problema de planejamento em IA. Esta abordagem é interessante devido à maturidade do planejamento em IA [65]. Além disso, nosso uso de anotações baseadas em ontologias a cada etapa do experimento científico forma a base para o rastreamento da proveniência dos dados e ferramentas [42]. Neste contexto, anotações são descrições associadas aos dados, serviços ou workflows, feitas pelo usuário, para melhor descrever os dados e experimentos.

As principais contribuições desta tese estão relatadas no restante do texto e detalhadas em textos adicionais, a saber:

- Proposta de uma solução para o problema da composição de serviços, combinando

resultados da IA e de Bancos de Dados, de forma a facilitar o projeto de workflows científicos e, ao mesmo tempo, documentar alternativas de projeto. Esta solução corresponde à especificação e implementação do Módulo de Composição Automática presente na Figura 1.1. Ela é parcialmente descrita no Capítulo 4 e está detalhada nas publicações [35, 37, 38];

- Definição de um modelo de dados que combine o armazenamento de workflows em camadas com o armazenamento de proveniência de dados e ferramentas. Este modelo é utilizado no Repositório de Workflows, Proveniência e Metadados na Figura 1.1 e é apresentado nos Capítulos 3 e 4 e detalhado na referência [10];

- Uso de repositórios de ontologias para enriquecer a semântica na construção automática de workflows e facilitar o rastreamento da proveniência de dados e procedimentos. O Repositório de Ontologias (camada Repositórios da Figura 1.1) é utilizado como entrada para o módulo de planejamento. O Capítulo 4 descreve este repositório. Informações adicionais sobre este repositório e sobre a relação do repositório com o planejamento são encontradas em [35, 36];

- Especificação de mecanismos de integração de dados, interoperabilidade de ferramentas e rastreabilidade de experimentos científicos. Utilizamos mecanismos de mapeamentos conceitual/de estruturas e casamento de interfaces para permitir a integração de dados e interoperabilidade. As regras que possibilitam esses mecanismos estão armazenadas nos Repositórios de Regras de Transformações de Dados e de Tipos Estruturais (camada Repositórios da Figura 1.1). Os dados de proveniência (armazenados seguindo as quatro camadas em que os workflows são definidos em nosso modelo) permitem a rastreabilidade dos experimentos. Os Capítulos 4 e 5 apresentam esses mecanismos e correspondem as publicações [34, 36];

- Validação da proposta pela implementação de três protótipos aplicados a diferentes domínios. O primeiro protótipo é focado na especificação automática e iterativa de experimentos em bioinformática. O Segundo trata questões de armazenamento e consultas a proveniência de experimentos e foi testado com dados biomédicos. O terceiro protótipo, centrado no projeto e execução de workflows via Web e parte dos mecanismos de integração e interoperabilidade, foi testado com dados e ferramentas oceanográficos. A Seção 6.3 descreve os protótipos implementados.

O trabalho desenvolvido na tese usou como base o WOODSS (WOrk-flOw-based spatial Decision Support System) [71], uma infra-estrutura computacional desenvolvida no Laboratório de Sistemas de Informação (LIS) [63] da Universidade Estadual de Campinas. Originalmente concebido para apoio a decisões em planejamento ambiental, ele evoluiu

para um ambiente extensível, centrado em bancos de dados que auxilia a especificação, reuso e anotação de workflows científicos e seus componentes [71, 90].

1.4 Organização da Tese

O restante do texto da tese está organizado como uma coletânea de alguns dos principais artigos publicados resultantes da pesquisa realizada. Cada artigo é apresentado na forma de um capítulo, com pequenas correções no texto original (padronização da nomenclatura, correção de erros ortográficos, etc). Todos os capítulos partem do uso de workflows científicos como forma de especificar experimentos em bioinformática.

O Capítulo 2 compara diversas infra-estruturas que gerenciam experimentos de bioinformática e apresenta a primeira versão da arquitetura para a infra-estrutura proposta. O Capítulo 3 descreve um modelo de dados em camadas para o armazenamento da proveniência de experimentos em e-Science. O Capítulo 4 apresenta uma versão aperfeiçoada da arquitetura, incluindo mecanismos de composição de serviços iterativa (semi-automática) e automática para a construção de workflows. O Capítulo 5 descreve os mecanismos de integração de dados, interoperabilidade e rastreabilidade de nossa infra-estrutura que facilitam o gerenciamento, compartilhamento e validação dos experimentos científicos. O Capítulo 6 apresenta as conclusões e possíveis extensões da tese.

1.4.1 Capítulo 2

O Capítulo 2 (*A framework based in Web services orchestration for bioinformatics workflow management*) contém os resultados publicados em [32] e é uma versão revisada do trabalho apresentado no Terceiro Workshop Brasileiro de Bioinformática [31].

Diversos sistemas foram desenvolvidos para tratar problemas de bioinformática (por exemplo, [29, 85, 96, 101]). Cada sistema possui um enfoque diferente, variando de sistemas específicos para um único projeto genoma até sistemas que visam integrar dados de todos os genomas seqüenciados.

O artigo compara sistemas desenvolvidos para auxiliar a execução de projetos genoma (principalmente na montagem e anotação de genomas). Com base nesta comparação, identificamos diversas características relevantes à especificação deste tipo de infra-estrutura de forma a permitir a integração de dados heterogêneos e prover interoperabilidade com demais sistemas e ferramentas. Dentre essas características, citamos: a disponibilidade dos serviços oferecidos, os mecanismos de integração de dados e a possibilidade de representação e execução de tarefas compostas.

Outro aspecto importante das infra-estruturas usadas para gerenciar projetos genoma é o conhecimento dos perfis de seus usuários. Destacamos três perfis de usuários comumente

encontrados em laboratórios de bioinformática: o *biólogo*, o *cientista da computação* e o *bioinformata*. Cada um destes perfis possui funções específicas num projeto genoma. Um sistema para gerenciamento de experimentos científicos deve se preocupar em fornecer ferramentas para auxiliar o trabalho de todos estes tipos de usuários.

Estes usuários podem ser classificados pela função que ocupam num projeto e por suas diferentes necessidades. Seguindo esses critérios, destacamos três categorias de usuários: *desenvolvedor de software*, *usuário-desenvolvedor* e *usuário final*. O desenvolvedor de software (*software developer*) é um cientista da computação ou um bioinformata que desenvolve as ferramentas básicas do sistema (por exemplo, ferramentas de alinhamento de seqüências de DNA, filtros, etc); o usuário-desenvolvedor (*user-developer*) é um cientista da computação ou um bioinformata que junta as ferramentas básicas (em workflows ou pipelines) utilizando scripts, programas ou com o auxílio de um sistemas gerenciador de workflows a fim de obter ferramentas mais complexas; o usuário final (*end user*) é, tipicamente, o biólogo que executa as ferramentas do sistema, analisa e anota os resultados mas não desenvolve novas ferramentas. Argumentamos que um sistema para gerenciamento de experimentos de bioinformática deve fornecer ferramentas que auxiliem essas três categorias de usuários.

Desta forma, o objetivo deste trabalho é a especificação e prototipagem de uma infra-estrutura para o gerenciamento e compartilhamento de experimentos científicos. Supomos que os experimentos de bioinformática podem ser representados como workflows científicos cujas unidades básicas são serviços Web. A infra-estrutura proposta possui ferramentas para auxiliar os três tipos de usuário: facilitando a integração de novas ferramentas básicas (serviços Web); provendo uma interface para especificação e execução de workflows científicos e possuindo mecanismos de busca (sintática e semântica) por serviços ou workflows disponíveis.

Mesmo contendo uma visão inicial da pesquisa desenvolvida nesta tese, este trabalho é interessante por comparar os sistemas existentes e por justificar nossas escolhas para o projeto da infra-estrutura. A evolução da infra-estrutura proposta pode ser acompanhada nos Capítulos 4 e 5, nos quais ela foi estendida para prover mecanismos de composição automática e semi-automática, apoio à proveniência de dados e ferramentas (utilizando o modelo apresentado no Capítulo 3) e rastreabilidade dos experimentos. Além disso, no Capítulo 5 estendemos os mecanismos de integração de dados e interoperabilidade de ferramentas. A Figura 1.1 mostra a versão final desta infra-estrutura, em alto nível.

1.4.2 Capítulo 3

O Capítulo 3 (*Automatic capture and efficient storage of e-Science experiment provenance* [11]) contém parte do trabalho realizado durante nosso primeiro estágio interna-

cional, realizado no grupo de bancos de dados [26] da Microsoft Research sobre orientação do pesquisador Roger Barga. O artigo sumariza o trabalho realizado durante o estágio e apresentado no evento *First Provenance Challenge* [10, 43]. Estas referências contêm detalhes do modelo de dados e de sua implementação.

Um aspecto importante dos experimentos científicos é a possibilidade de se descobrir a proveniência de cada uma das informações utilizadas no experimento. Este tipo de informação, além de garantir a qualidade de um experimento, também é fundamental para que este possa ser reproduzido por outros grupos.

Tipicamente, os dados de proveniência são gerados por anotações dos experimentos, feitas pelos cientistas sem a existência de ferramentas de apoio. Esta abordagem, além de ser muito laboriosa, dificulta o entendimento dessas anotações por terceiros, pois elas nem sempre são estruturadas e não costumam compartilhar um vocabulário.

Como os experimentos científicos podem ser vistos como workflows científicos na e-Science, a proveniência dos experimentos corresponde ao registro das atividades que foram executadas por um dado workflow, incluindo serviços e bancos de dados acessados, conjuntos de dados usados, parâmetros utilizados, etc. Argumentamos que este tipo de proveniência deve ser automaticamente produzido pelo sistema gerenciador de workflows e gerenciada por um sistema subjacente de armazenamento.

Diferentes tipos de experimentos possuem necessidades distintas de armazenamento de proveniência. Desta forma, um modelo único e não flexível de armazenamento de proveniência não seria capaz de satisfazer a todos os tipos de requisitos.

O objetivo do artigo é propor um modelo flexível e em camadas para o armazenamento da proveniência da execução de workflows. Este modelo é dividido em quatro camadas: Workflow Abstrato (*Abstract Workflow*), Modelo de Workflow (*Workflow Model*), Workflow Executável (*Executable Workflow*) e Workflow em Tempo de Execução (*Runtime Workflow*). Cada uma destas camadas armazena informações sobre o workflow propriamente dito, além de informações sobre proveniência dos dados e ferramentas. A camada Workflow Abstrato armazena informações sobre a estrutura do workflow (atividades abstratas e ligações entre essas atividades). O Modelo de Workflow contém informações sobre a instanciação das atividades (substituição das atividades abstratas por concretas). A camada Workflow Executável armazena informações sobre dados de entrada e parâmetros das atividades. A camada Workflow em Tempo de Execução contém informação sobre a execução do workflow: quando cada atividade foi executada, onde, quais os dados gerados, exceções que ocorreram, etc. Além dos dados de proveniência produzidos e armazenados pelo sistema gerenciador de workflows, o usuário pode inserir anotações, dentro de qualquer camada, sobre os dados, as ferramentas e os workflows. O trabalho foi validado com a construção de protótipo usando o sistema gerenciador de workflows Windows Workflow Foundation [8] e SQL Server 2005.

1.4.3 Capítulo 4

O Capítulo 4 (*An ontology-based framework for bioinformatics workflows*) contém os resultados publicados em [36].

Este trabalho apresenta uma infra-estrutura baseada em ontologias para o projeto, reuso, anotação e documentação de workflows em bioinformática. As principais contribuições em relação ao trabalho apresentado no Capítulo 2 são os mecanismos de composição manual, iterativa e automática combinando uso intensivo de ontologias e planejamento em inteligência artificial.

Ontologias são usadas como base para a anotação de dados e serviços. Além disso, as relações entre os conceitos das ontologias são utilizadas para a verificação semântica da compatibilidade de dados e serviços.

Os repositórios da infra-estrutura armazenam, além dos dados, workflows anotados por ontologias e as ontologias propriamente ditas. Há duas ontologias: *Ontologia de Domínio* e *Ontologia de Serviços*. A Ontologia de Domínio contém conceitos de bioinformática (por exemplo, *seqüência de DNA*, *gene* e *genoma*). A Ontologia de Serviços contém os tipos de serviços (por exemplo, *serviço de alinhamento*, *serviço de montagem* e *filtros*). As instâncias dos serviços são armazenadas no *Catálogo de Serviços* que estende as funcionalidades de um UDDI (Universal Description Discovery and Integration) por armazenar ligações entre os serviços e as ontologias. Cada operação de um serviço é anotada como sendo de um tipo (da Ontologia de Serviços) e cada parâmetro da interface desta operação é anotado seguindo a Ontologia de Domínio.

A composição de atividades para a construção de workflows é realizada utilizando *planejamento* em Inteligência Artificial. A arquitetura foi validada pela implementação de um protótipo. Comparamos alguns sistemas de planejamento e justificamos nossa escolha pela ferramenta SHOP2. Estendemos a definição de domínio de SHOP2 para que ela se beneficie dos relacionamentos ontológicos de nossas duas ontologias de forma a melhorar a qualidade dos planos (workflows) produzidos. Detalhes adicionais da implementação são descritos em [35, 37, 38].

O artigo descreve os três tipos de composição: *manual* (supervisionada), *iterativa* (ou semi-automática) e *automática*. Cada tipo de composição é mais adequada às necessidades de cada perfil de usuário. Um usuário com maior conhecimento sobre as ferramentas poderá preferir compor manualmente os workflows. Por outro lado, um usuário que não conheça as características de cada ferramenta preferirá que o sistema gere automaticamente um workflow que satisfaça seus objetivos.

1.4.4 Capítulo 5

O Capítulo 5 (*Traceability Mechanisms for Bioinformatics Scientific Workflows* [34]) contém os resultados publicados no AAAI2007's Workshop on Semantic e-Science (SeS07).

O trabalho combina o modelo de proveniência apresentado no Capítulo 3 somado com as funcionalidades da infra-estrutura proposta no Capítulo 4 para juntar a especificação automática de workflows e o armazenamento da proveniência de seus dados e ferramentas. O uso de ontologias e de serviços da Web Semântica facilita o compartilhamento e reuso de serviços distribuídos na Web. Porém, na prática, ainda restam alguns empecilhos ao trabalho científico. Estes empecilhos estão relacionados a: *integração de dados*, *interoperabilidade* e *rastreabilidade*.

O uso de ontologias ajuda a estabelecer um vocabulário consensual sobre os dados e as ferramentas de um certo domínio de aplicação. Porém, dados referentes a um mesmo conceito de uma ontologia podem estar representados em estruturas diferentes. Desta forma o uso de ontologias não resolve, por si só, o problema de integração de dados. Há diversas soluções para a integração de dados heterogêneos [50], variando do estabelecimento de um modelo de dados consensual até registros de mapeamentos entre diferentes estruturas de dados. Neste trabalho, combinamos o mapeamento entre estruturas e nossas ontologias para facilitar a integração de dados.

Rastreabilidade significa a habilidade de descrever a maneira em que algum produto foi desenvolvido, incluindo todos os processos pelos quais ele passou. A implementação de mecanismos de rastreabilidade depende de dois fatores: (i) o armazenamento da informação relevante de cada experimento e (ii) o fornecimento de ferramentas para o acesso, de maneira recursiva, de todos os processos envolvidos na geração de um certo dado ou workflows. Este artigo estende os mecanimos de armazenamento e gerenciamento de dados proposto no Capítulo 4 de forma a facilitar a rastreabilidade e descreve algumas das ferramentas propostas para permitir a rastreabilidade de experimentos. Além disso, ele discute como o conhecimento do domínio pode ser usado para melhorar os mecanismos de rastreabilidade.

Outro conceito explorado para possibilitar a integração de dados e a interoperabilidade entre ferramentas é a avaliação da compatibilidade entre estruturas de dados ou interfaces de serviços. Este trabalho partiu do trabalho de Santanchè et al [88] sobre compatibilidade de interfaces e classificou cada transformação entre duas estruturas (ou interfaces) segundo dois critérios: *precisão* e *equivalência*.

As principais contribuições apresentadas neste capítulo são os mecanismos para facilitar integração de dados, interoperabilidade de serviços e rastreabilidade de experimentos. Esses mecanismos são providos por um módulo de gerenciamento de dados chamado *Data Manager* que torna transparente aos demais módulos o esquema de armazenamento de cada informação.

1.4.5 Outras publicações

Além dos artigos que compõem o corpo desta tese, outros trabalhos foram publicados como parte desta pesquisa. Eles estão listados a seguir, cronologicamente.

- *A data model for comparative genomics* [30] publicado na Revista Tecnologia da Informação. Este artigo apresenta um modelo de dados para ser utilizado em sistemas de informação para genômica comparativa. O modelo proposto é extensível e facilita a descoberta de relações entre organismos e famílias gênicas. Validamos o modelo construindo um banco de dados populado com oito genomas de organismos procariotos. Esta validação demonstrou a utilidade do modelo para a integração e comparação de dados sobre genes.

- *Genome features of Leptospira interrogans serovar Copenhageni* [79] publicado no Brazilian Journal of Medical and Biological Research. Artigo na área de biologia que relata um conjunto de características encontradas no genoma da bactéria *Leptospira interrogans serovar Copenhageni*. Este artigo não está diretamente ligado às contribuições desta tese, porém a pesquisa relatada nele colaborou em dois aspectos importantes. O primeiro foi um melhor conhecimento da área de montagem e anotação de genomas devido ao desenvolvimento de ferramentas para gerenciar os dados genômicos. O segundo aspecto foi um melhor entendimento dos problemas e requisitos dos biólogos e bioinformatas ao utilizarem essas ferramentas.

- *Comparative analyses of Xanthomonas and Xylella complete genomes* [77] publicado na revista OMICS: A Journal of Integrative Biology. Este artigo, também na área de biologia, relata a comparação entre genomas de organismos de dois gêneros: *Xanthomonas* e *Xylella*. Para a realização deste trabalho foi necessário o desenvolvimento de ferramentas de integração e comparação de dados. Nosso modelo de dados para genômica comparativa [30] foi utilizado para o estabelecimento de algumas relações entre os genomas.

- *Fact and Task Oriented System for genome assembly and annotation* [39]. Este resumo estendido foi apresentado no Simpósio Brasileiro de Bioinformática em 2005 e posteriormente publicado na série *Lecture Notes in Bioinformatics*. Ele divide o trabalho de montagem e anotação de genomas em fatos (*facts*) e tarefas (*tasks*). Os fatos correspondem ao conhecimento do genoma (dados) e as tarefas são as operações que devem ser executadas para se obter informações adicionais sobre os dados. Este trabalho se preocupa com aspectos de usabilidade e visualização de dados a fim de facilitar a interação com o usuário e permitir um uso mais sistemático das ferramentas.

- *WOODSS and the Web: Annotating and Reusing Scientific Workflows* [71] publicado na revista SIGMOD Record. Este artigo apresenta 7 anos de pesquisa em workflows científicos realizada no LIS - UNICAMP. Como resultado desta pesquisa foi desenvolvido (e está sendo estendido) um sistema chamado WOODSS [71, 90]. Os dois principais objetivos desse sistema são: ajudar cientistas a especificar, reusar e anotar seus modelos e experimentos; e documentar os esforços colaborativos nas atividades científicas.

- *Evaluation of graph based protein clustering methods* [27] apresentado no Quinto Simpósio Brasileiro de Matemática e Biologia Computacional. Este artigo propõe uma metodologia para avaliar agrupamentos de proteínas. Esses agrupamentos são amplamente utilizados para caracterizar funcionalmente as proteínas. Há diversas soluções para se tratar este problema, dentre as mais usadas estão as baseadas em grafos. Neste artigo revisamos diversas propostas baseadas em grafos e apresentamos uma metodologia para avaliar os resultados destas propostas.

- *A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows* [37]. Este relatório técnico apresenta parte da infra-estrutura proposta nesta tese. Além de detalhes da arquitetura ele contém exemplos de parte da definição do domínio em SHOP2 (utilizada durante o planejamento). Uma versão revisada e reduzida deste trabalho foi posteriormente publicada [36] e corresponde ao Capítulo 4 desta tese.

- *Bioinformatics scientific workflows: combining databases, AI and Web services* [38]. Este artigo, apresentado no Workshop de Teses e Dissertações de Bancos de Dados em 2006, sumariza o trabalho realizado nesta tese até maio de 2006.

- *Automatic Generation of Workflow Provenance* [9]. Este artigo, apresentado em International Provenance and Annotation Workshop (IPAW2006), argumenta que a proveniência de dados deve ser gerada automaticamente pelo sistema gerenciador de workflows e deve ser mantida por um sistema subjacente de proveniência. Este trabalho apresenta um modelo flexível e em camadas para o armazenamento de proveniência de workflows. A validação do modelo foi feita por meio de um protótipo que une o modelo de proveniência proposto ao modelo XML de representação de workflows da Microsoft [8]. O Capítulo 3 desta tese possui a versão revisada e estendida deste modelo.

- *A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows* [33]. Este resumo de 3 páginas foi apresentado no Segundo Workshop de Teses de Doutorado do Instituto de Computação

da UNICAMP em 2006 e sumariza o trabalho desenvolvido na tese até o início do segundo semestre de 2006.

- *GeneProjects: a Web application for ongoing annotation in EST and Shotgun genome projects* [20] aceito para publicação na revista Genetics and Molecular Biology. Este artigo descreve um sistema que foi desenvolvido no Laboratório de Genômica e Expressão (LGE) [62] do Instituto de Biologia da UNICAMP. Este sistema tem como objetivo o gerenciamento de projetos genoma e a identificação de características biológicas a partir de seqüências de DNA. Em particular, ele se destaca dos outros trabalhos que possuem objetivos semelhantes por ser focado no chamado *projeto*. Um projeto consiste de uma pesquisa específica dentro do genoma, tipicamente para se encontrar um particular gene ou via metabólica.

- *AI Planning in Web Services Composition: a review of current approaches and a new solution* [35] apresentado no Sexto Encontro Nacional de Inteligência Artificial (VI ENIA) compara diversos planejadores utilizados para a composição de serviços Web. Além disso, o artigo descreve a combinação do uso de SHOP2 e do repositório de ontologias para aperfeiçoar o processo de produção de planos.

Além disso, seis pôsteres também foram produzidos durante a tese. Dois deles foram apresentados nos eventos UNICAMP de Portas Abertas (UPA) de 2004 e 2005. Estes pôsteres contêm informações sobre as pesquisas realizadas no Laboratório de Bioinformática [61] do Instituto de Computação da UNICAMP, principalmente dados sobre montagem e anotação de genomas. O terceiro pôster foi apresentado no evento *International Workshop on Genomic Databases* (IWGD'05) no ano de 2005 e destaca os mecanismos de composição de serviços (manual, iterativa e automática) para a construção de workflows. O quarto pôster, entitulado *GeneProjects: a Web application for ongoing annotation in EST and Shotgun genome projects*, foi apresentado no evento *1st Internacional Conference of the Brazilian Association for Bioinformatics and Computational Biology* em 2005 e destaca as principais funcionalidades do sistema para o gerenciamento de projetos genoma chamado *GeneProjects*. Os dois últimos pôsteres serão apresentados no evento *International Workshop on Genomic Databases 2007* e descrevem, respectivamente, o novo protótipo de nossa arquitetura voltado para o projeto e a execução de workflows via Web e novos caminhos para a rastreabilidade e anotação de experimentos em bioinformática.

Capítulo 2

A framework based on Web services orchestration for bioinformatics workflow management[2]

2.1 Introduction

Bioinformatics activities are growing all over the world. Among the various problems, there is the question of providing a framework for inter-institutional cooperation. One of the directions considered is to use the new service technologies - Web services [4] and Grids [44].

Web services are a good approach to solve heterogeneity problems.The use of XML[105] and standard Internet protocols contribute greatly to popularization and dissemination of Web services. Important issues are service and data discovery, service execution and coordination. Thus, there is a need for management mechanisms for data and services, and for supporting enhanced semantics.

The main goal of our work is to propose and develop a framework to solve some of these problems, for bioinformatics applications. In this specific application domain, there are already some incipient proposals that involve the coordination of distributed tasks by using workflows [19, 49, 73, 103] and Web services [53, 105]. These proposals suffer from problems described previously; moreover, there is a lack of standards for interfaces among tools used by end-users. Thus, besides contributing towards managing data and services, the framework will contribute to help tool interoperability.

The expected results are the specification and development of a framework for bioin-

[2]**L.A. Digiampietri, C.B. Medeiros, and J.C. Setubal. A framework based in Web services orchestration bioinformatics workflow management.** ***Genetics and Molecular Research*****, 4(3):535–542, 2005.**

formatics applications capable of:

- specifying workflows, via composition of Web services, and storing these specifications;
- discovering services and workflows of interest, in a semantic way;
- managing workflow execution via service orchestration and
- auditing workflow execution.

The rest of this text is organized as follows. Section 2.2 shows related work. Section 2.3 describes the main aspects of the proposed framework. Section 2.4 contains conclusions and ongoing work.

2.2 Related work

2.2.1 Systems and frameworks

Many works consider the integration of bioinformatics data and tools. Some emphasize functionality for a specific research team whereas others concentrate on supporting cooperation on the Web, for teams within a given project. The main goals of these systems, summarized in Table 2.1 are:

- to provide a set of bioinformatics tools,
- to allow data and tool integration and
- to build a framework for one specific bioinformatics project.

Systems that provide a set of bioinformatics tools make their tools available via Web sites and/or via a local program [12, 40, 49, 101]. Usually these tools are developed under specific standards, hampering the integration of tools built by different groups. The framework of Eckart and Sobral [40] employs Web services in the server side and an application on the client side. When the application is started, the list of available services is updated, allowing clients to invoke new services. Standard inputs and outputs allow the output of a service to be used as input of another. This framework does not yet allow automatic integration of tools.

Several systems provide some level of data integration. Some have the goal of integrating large volumes of available genomic data in one generic data model [85]. Other systems aim the modeling of any genomic project via a set of basic components [96].

Characteristic	BioOpera[12]	Source[29]	Hall[49]	CMR[85]	GGB[96]	myGrid[101]
1- execution of a task in a distributed environment	x		x			x
2- maintenance of repository of bioinformatics tools		x			x	x
3- provide some level of tool integration	x	x	x	x	x	x
4- modeling workflows of a complex task	x		x			x
5- multi-institutional sharing of resources	x	x	x	x	x	x
6- multi-institutional development of tools						
7- coordination of workflow execution	x		x			x

Tabela 2.1: Some surveyed systems and their characteristics.

Finally there are systems that link several kinds of services and data to facilitate the genomic annotation of one specific genome project [29]. Some systems handle the problem of tools integration. The specification of tasks interaction and interdependencies relations are typically designed using workflows [12, 49, 101]. The problem lies in workflow specification and execution.

There are two main kinds of frameworks for bioinformatics projects. In the first kind, all tools are developed for a specific project [61]. Whenever a new genome project is started, scientists need to adapt the entire framework. The second kind contains the frameworks formed by basic components [96]. The framework of each new genome project is constructed by combining components with low configuration costs. Both kinds of framework are especially good for genomic assembly and annotation of specific genomes, but their tools can not be accessed by other projects.

Our framework differs from the surveyed related work in the following ways. First, it integrates all 7 characteristics of Table 2.1. Second, it allows user interaction while tools are executing. Third, it focuses multi-institutional development of tools using Internet standards. This means that these tools are available not only for one project but to any project that complies with these standards. Finally, tools are managed via service orchestration.

2.2.2 Related issues

Related work involves research on Web services and their orchestration, scientific workflows and bioinformatics tools and data.

A Web service is “a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A

Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols" [105]. Some open topics are how to discover adequate services and service providers, how to automate Web services integration, how to minimize semantics ambiguity in services specifications and how to assign information about quality and reliability of the services offered by a provider [4]. Our work concentrates on the aspects of specification of interfaces for bioinformatics services and their orchestration.

Service orchestration is a centralized mechanism that describes how diverse services can interact. This interaction includes message exchange, business logics and order of execution. The most important works in the coordination of Web services involving BPEL4WS [7], OWL-S [23] and WSCI [104]. We will adopt BPEL4WS as a basis for specifying service orchestration.

A workflow denotes the controlled execution of multiple tasks in an environment of distributed processing elements. Workflows represent a set of activities to be executed, their interdependencies relations, inputs and outputs [90].

Bioinformatics workflows are scientific workflows, i.e., they differ from a usual workflow because they have some additional characteristics like high degree of flexibility, uncertainty and existence of exceptions [106].

Our work is concentrated in the execution of scientific workflows through the orchestration of Web services and user interaction. Open problems that will be attacked include communication protocols and interfaces among services to specify a workflow.

There are many tools and databases for bioinformatics. Samples of tools include BLAST [6], Phred [41] and Consed [48]. These tools are geared towards sequence comparison, analyze and visualization. Other complex problems with dedicated tools involve fragment assembly of DNA (alignment and consensus), phylogenetic trees, database search, etc. Our research will concentrate on the applications for assembly, annotation and comparison of genomes. The choice of these applications was based in prior experience of the Laboratory for Bioinformatics (LBI) [61] at UNICAMP in assembly and genomic annotation. This choice is also common to several bioinformatics efforts using workflows [73], clusters [12] and grids [101].

2.3 The proposed framework

The framework will manage design and execution of scientific workflows that will support execution of distributed bioinformatics applications on the Web.

One problem faced by scientists in such a context is integrating these procedures via adequate interfacing among tools. Our approach handles this problem by encapsulating data and tools by Web services. Figure 1 shows how the framework will support the

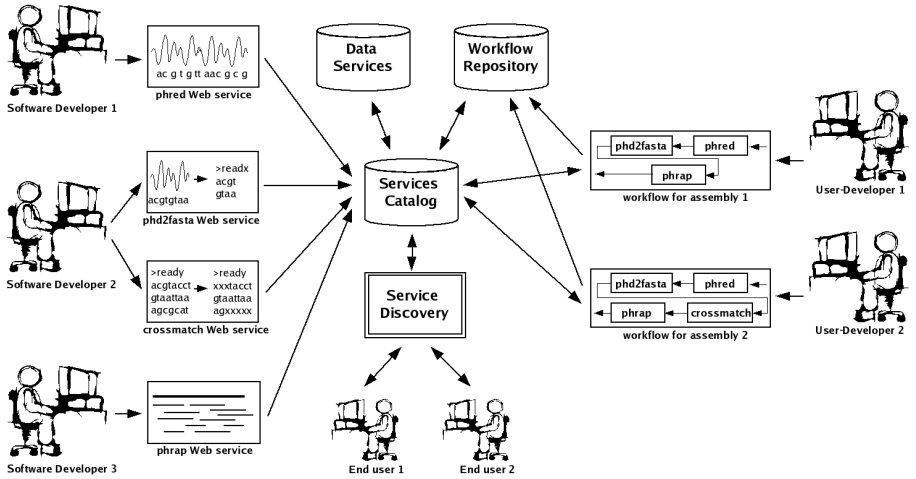


Figura 2.1: User interaction overview

main user activities in assembly of genomic data. There are three kinds of users that interact with our framework: software developer, user-developer and end user. Software developers design Web services and subscribe these services to the Service Catalog. Users-developers use our framework to design workflows that determine how complex tasks must be composed and executed. End users invoke a Web service or a Workflow designed by a user-developer. For instance, software developer 1 develops a phred Web service that is stored in the service catalog. User-developer 1 specifies an assembly workflow that invokes this phred service and is stored in a specific repository. When end user 1 requests execution of some assembly task, the service discovery module will inform the user there are 2 available workflows, and the user can then choose which workflow to execute. Workflow activities embed tools that are executed via services; data is also made available via services.

Figure 2.2 shows our system architecture that supports the integration of the tasks shown in Figure 2.1.

The Service layer manages the bioinformatics Web services that must provide basic operations such as assembly, matching and consensus, creation of descriptors and genomic annotation. They are amenable to composition to provide more sophisticated functionalities - e.g. genomic comparison and gene family operations. Here, we started by transforming modules already available in LBI into services.

The Service catalog layer is responsible for storing Web services' syntactic and semantic descriptions, as well as the URI where each service can be found. This layer will utilize a schema of subscription / unsubscription to register the services. It must maintain a history of services availability and allow reuse of workflows.

Service discovery can be done in several ways. Our framework will allow search by

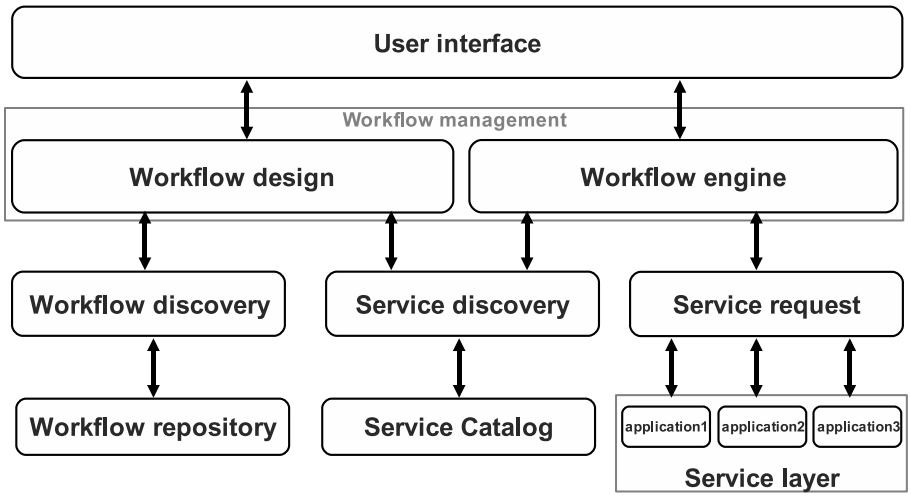


Figura 2.2: Framework architecture.

functionality, context and syntax. Search by functionality and context will be done based in semantic data (metadata) assigned to the services. Search for compatible syntax will be based on the parameters of the service interface. These search methods will use techniques already discussed in the literature [21, 100] about syntactical, ontological and semantic matching.

The Service request layer will be responsible for the management of each Web service solicitation. This layer communicates with the Web services provider, sending input data and receiving results. It is responsible for detecting service failure such as unavailable service or time limit violation. The Workflow engine layer is responsible for the controlled execution of all workflow tasks, via orchestration. The operation functions provided by the Workflow engine are interpretation of the process (or task) definition, creation and management of process instances, navigation between activities and supervisory and management functions [24].

The Workflow design layer must support workflow specification and edition. The facilities provided are: graphical interface for workflow edition, service list, interface description of selected services and syntactical check. It will use the scientific workflow editing tools developed at UNICAMP [90].

Our framework is being specified and developed using a bottom-up approach. We started with bioinformatics basic services specification and development encapsulating LBI tools into services e.g. genomic annotation and comparison tools. This stage is also establishing the metadata types that must be associated with the services. The strategy for this stage requires initial definition of some bioinformatics basic services.

The second stage will be the study and development of techniques for service discovery and request using syntactic and semantics search mechanisms.

The following step involves the specification and development of methods for workflow design and execution. Each workflow activity is a service or a bioinformatics application. This stage will make use of existing work on management of scientific and distributed workflows [58, 25] and tools developed at UNICAMP [90]. Here it will be necessary to specify and to implement an orchestration mechanism for these kinds of services (specific to workflows). Workflows data sources and providers will be encapsulated by services.

System tests will be based on large volumes of real data from LBI.

2.4 Conclusions and ongoing work

The main contribution of this work is the framework itself. It will allow multi-institutional cooperation via data, tools and workflow sharing. Various kinds of users will be able to interact with our system and with each other to achieve a given goal. Other contributions lie in the solution of open problems in scientific workflow specification via composition of Web services and semantic specification of bioinformatics tasks. Another important contribution is the methodology for integration of these solutions for bioinformatics.

The work accomplished so far can be divided into research and practical work. The research was concentrated in the analysis of related work and tools utilized by bioinformatics research centers. The practical work was concentrated on development and utilization of LBI assembly and annotation genomic systems. Furthermore, we modeled and implemented a comparative genomic system [30]. These activities allowed the understanding of bioinformatics applications in terms of types of data and applications involved. At this moment, we are specifying the Web service semantic description and encapsulating LBI tools.

Capítulo 3

Automatic capture and efficient storage of e-Science experiment provenance[3]

3.1 Introduction

Scientific workflows are now recognized as a crucial element of the "cyberinfrastructure", facilitating e-Science. Typically sitting on top of a middleware layer, scientific workflows are a means by which scientists can model, design, execute, debug, re-configure and re-run their analysis and visualization pipelines. Part of the established scientific method is to create a record of the origins of a result, how it was obtained, experimental methods used, machine calibrations and parameters, etc. It is the same in e-Science, except provenance data is a record of the workflow activities invoked, services and databases accessed, data sets used, and so forth. Such information is useful for a scientist to interpret their workflow results, and for other scientists to establish trust in the experiment result.

In this paper, we describe how a workflow enactment engine can assume responsibility for generating workflow provenance automatically during runtime [14]. We argue that a single representation can not satisfy all provenance queries. In some cases, it is suitable to provide an abstract description of the scientific process that took place, without describing specific codes executed, data sets or remote services invoked. In other cases, a precise description of the execution is exactly what is required to explain a result, including all operational details such as parameters and machine configurations. We present a provenance model that supports multiple levels of representation [71, 84], that allows

[3] **R.S. Barga and L.A. Digiampietri. Automatic capture and efficient storage of e-Science experiment provenance. *Concurrency and Computation: Practice and Experience*, 2007. Accepted for publication.**

users to navigate from the abstract levels into lower levels and back again, and permits scientists to specify exactly what they wish to share from their experiment.

We are developing a system called REDUX to explore the benefits and challenges of automatically capturing experiment provenance, along with methods to store efficiently the resulting provenance data. REDUX will enable scientists to recall or restore provenance data to understand or explain an experiment and establish trust in the result. REDUX uses the Windows Workflow Foundation (WinWF) [8] as workflow engine. This paper presents a current snapshot of our project, including an overview of our model for experiment provenance, implementation in a commercial workflow system, and techniques to store provenance data efficiently in a relational database. Our system addressed the challenges presented in the First Provenance Challenge [76] and was able to answer all the proposed queries. Moreover a comparison with the approaches of the other participating teams allowed us to identify several ways in whose our system can be improved. This discussion can be found in Sections 3.4 and 3.5.

Clearly, we can not present a complete system description in this paper; however there are a number of concrete contributions to report on at this point in time. We present our model for workflow execution provenance, which consists of layers from an abstract description of the workflow (experiment design) to a execution details from an experiment run. We describe the storage of provenance data generated from our model on a relational database system. To make the storage of provenance data economically feasible, we identify properties of provenance data represented by our model that can significantly reduce the amount of storage required.

3.2 Workflow Execution Provenance Model

In this section we first motivate and then describe our model for representing result provenance captured during workflow execution.

3.2.1 Different Kinds of Result Provenance

We frame our discussion using the Provenance Challenge workflow example. A scientist is at the center of an experiment as it unfolds, possibly over a period of days or weeks, interacting with the executing processes. This scientist may alter the parameters of *align_warp*, navigate between databases available online to issue queries, inject "shim" code as needed (data transformations) in order to get the output from one step (workflow activity) to feed into the next step (workflow activity). The experiment (workflow) is specified in the abstract as a schedule of activities (warp an input image, followed by some other image transformations, followed by a image slicer activity, followed by a user-

defined image converter), then instantiated with concrete services align_warp → reslice → softmean → slicer → convert) and invoked with specific parameters at runtime by the scientist (align_warp version 5 with parameters `-m 12` and `-q` → reslice version 5 without parameters → softmean with parameter `y null` → each instance of slicer with parameters, respectively, `-x .5`; `-y .5`; and `-z .5` → convert without parameters, producing a gif image), and dynamically altered if the scientist observes an anomaly (he used wrong reference images as input of align_warp, or he wants to change the value of a parameter, so the scientist alters some of the parameters and reruns the activity before continuing on).

Later, perhaps after several weeks, the scientist may ask the following questions about an experimental result:

- Which version of *align_warp* did I use?
- What codes (activities) did I invoke during workflow execution, and what were the parameters?
- What machine was used to execute the *softmean*?

A simple *derivation path* that records the workflow execution, identifying workflow instance, input variables and runtime parameters, would suffice to answer these provenance queries. However, other information can be associated with a result to provide additional context that more accurately explains the experiment execution and its result. Consider the following provenance queries:

- What experiments utilized *image converter* tools?
- Were any steps skipped in this experiment? Were any shims (data transformations) inserted?
- Did the experiment design differ between these results, and if so what were the differences?
- Are there branches in the experiment that have not been explored yet (executed) by some experiment?

These queries illustrate the need for provenance information captured at the *experiment design level* rather than simply the derivation path level. Both types of provenance information communicate context necessary to understand or explain an experiment and establish trust in the result, but how this information is captured, represented and queried is likely to be quite different. To capture the experimental process, we must adopt a more general notion of result provenance.

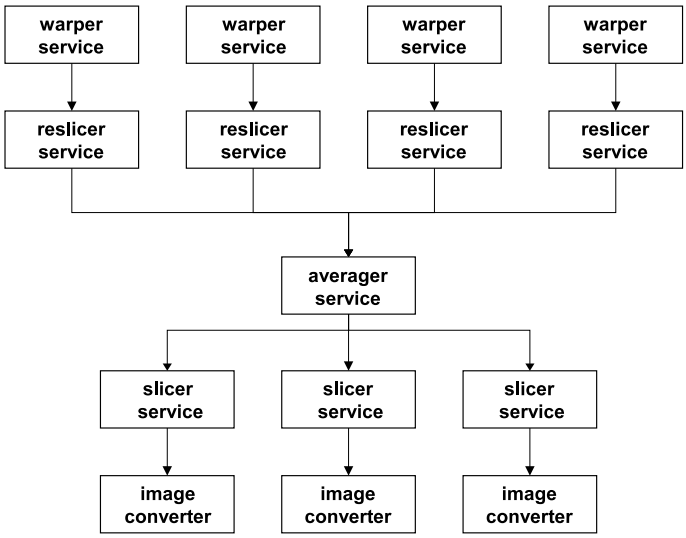


Figura 3.1: L0: Abstract service descriptions

We argue the complete provenance for an experiment result is captured by a set of objects to which it is incrementally bound. This process begins with an abstract description of the workflow (experiment), before it is bound to specific services, tools, data sets, etc, and continues until the execution of the experiment. Intuitively, our layered provenance model holds some of these objects fixed while varying others. To capture flexibly the degrees of freedom in which an experiment can vary, without needlessly duplicating objects that stay fixed, we use *binding levels* to model our observation that experiment design (abstract workflow), a workflow instance, and even individual processing steps (workflow activities) acquire their identity in stages. In Section 3.2.2, we present our layered model for result provenance, in which each level represents a distinct stage of binding.

3.2.2 A Layered Model For Result Provenance

The first level in our model, illustrated in Figure 3.1, represents an abstract description of the experiment. This layer captures *abstract activities* in the workflow and *links/relationships* among the activities or ports of activities, where an abstract activity is a *class* of activities. The provenance data for L0 supports general queries, such as: *what experiments in my collection use web services?* In general, level L0 describes the design space workflow instances of the abstract model.

The second level in our model, illustrated in Figure 3.2, represents an instance of the abstract model. This layer captures bindings or *instances of activities* and additional relationships, as classes of activities are instantiated. This level refines abstract activities specified in the previous level with specific instances (for example, the abstract activ-

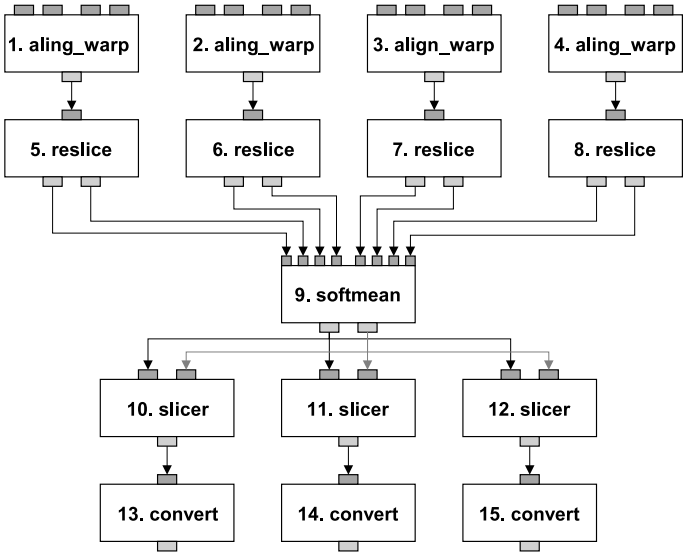


Figura 3.2: L1: Service instantiation descriptions

ity *Image Converter tool* is bound to the instance *convert tool*). From the information captured in Level L1 a user can pose provenance queries about specific activities: *What experiments (workflows) used the service convert?*

The next level of our model L2, illustrated in Figure 3.3, represents the execution of the workflow instance specified in level L1. From level L1 to level L2, the model captures information provided at runtime that complete the specification of activities, including input data, parameters supplied at runtime, branches taken, activities inserted or skipped during execution, etc. Information captured at this level is sufficient to trace the execution of the workflow. In level L2, data sets and parameters are captured as the workflow is executed. From this, we can pose a number of queries on how the experiment was actually carried out. Such queries can involve parameters, data values supplied at runtime, which activities were inserted or skipped, etc. Moreover, if the workflow enactment engine provides the necessary support, this provides sufficient information to serve as a redo log for the smart replay of the experiment. Our smart replay has functionality that allows the user to select what activities will be re-executed and what activities will use the stored result (produced in the previous execution of the workflow, stored in L3).

The final level of our model L3, illustrated in Figure 3.4, represents runtime specific information. Here we capture operational details, such as the start and end time of workflow execution, start and end time of individual activity execution, status codes and intermediate results, information about the internal state of each activity, along with information about the machines to which each activity was assigned for execution.

Level L3, the runtime operational level, can contain a diversity of information. The

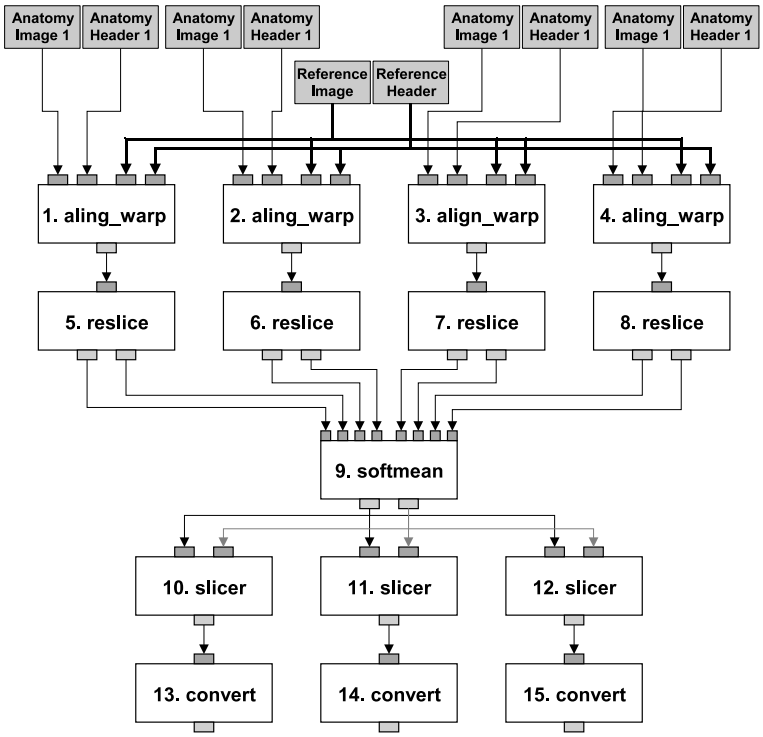


Figura 3.3: L2: Data instantiation of workflow

kind of information to be stored will depend of the goals of the user. For example, in a system that utilizes a high performance Grid, the user may wish to record the processor in the Grid on which the job was executed and when execution occurred. In other systems, the user may wish to store information about the internal status of each activity. These examples are supported by our provenance model, but they require cooperation from external software. In the Grid example, the scheduler must send job information to the provenance system, while in the second example each activity must send activity status information to the provenance system.

3.2.3 Provenance Model Summary

To sum up, the benefits a multilayered model can bring are *explanation*, *validation*, *insight* and *control*. Having an abstract model of the science being undertaken and being able to use this model to interpret behavior (services and data sets used), together make it possible to explain or reason about an experiment. For validation, knowing which individual activities were executed, identifying branches taken during execution, steps skipped or inserted, and parameters supplied at runtime are essential to establish trust in a result. The ability to compare a set of related experiment executions against an abstract

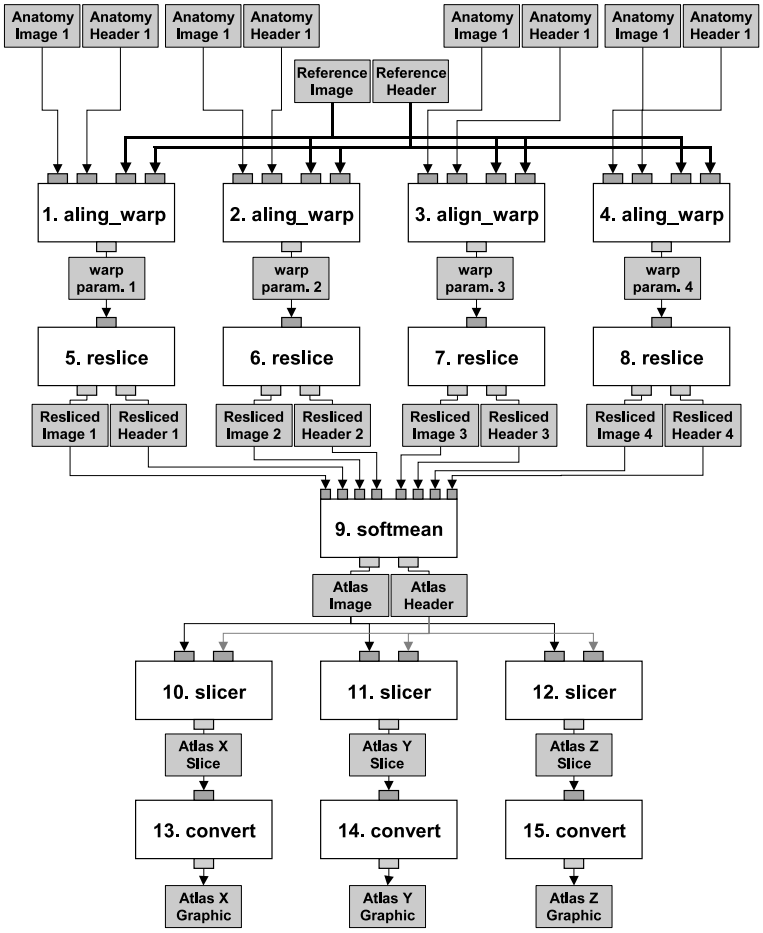


Figura 3.4: L3: Run-time provenance from execution

workflow model, to identify common patterns or options not yet explored, can be used to gain insight in authoring new experiment designs. Finally, a layered model offers the scientist control over information they wish to share. A related benefit discussed later in the paper is optimization - a layered or factored model can expose opportunities to store efficiently provenance traces in a storage manager.

3.3 Storing Provenance Data

In this section we consider how to store experiment result provenance. We believe a database management system is a promising candidate. A database stores data reliably, allows indices to be built on top of provenance data, and has a rich query interface that can be used to pose provenance queries. And, since the database is a separate entity from the workflow management system, the persistence of the result provenance is independent of the experiment (workflow).

This leads to an obvious question – *what database schema is best suited for storing result provenance?* To answer this, we first consider typical provenance queries, using queries from the *First Provenance Challenge* [43, 76]. Then we consider advantages and disadvantages of various schemas to record provenance. Finally, we consider issues related to efficiently storing provenance.

3.3.1 Result Provenance Schema Discussion

Our provenance schema design was driven in part by considering provenance queries, in particular queries from the *First Provenance Challenge*. The goal of the challenge was to foster understanding the expressiveness of the capabilities of provenance-related systems, including how each system answers a set of nine provenance queries. Some of the provenance queries in the challenge [43] are as follows:

FPC #4: Find all invocations of procedure *align_warp* using a twelfth order nonlinear 1365 parameter model that ran on a Monday.

FPC #6: Find all output averaged images of $softmean$ procedures, where the warped images taken as input were *align_warped* using a twelfth order nonlinear 1365 parameter model, i.e, where $softmean$ was preceded in the workflow by *align_warp* procedure with argument -m 12.

FPC #8: A user has annotated anatomy images with a key-value pair $center = UChicago$. Find the outputs of the *align_warp* activity where the inputs were annotated with $center = UChicago$.

Based on these queries, we can see the result provenance data that needs to be recorded can be broadly classified into the following two categories:

Dependency Provenance: This refers to provenance used to record the fact that one computational step, or activity, either *depends on*, or was *derived from*, information computed in another step. For example, *align_warp* precedes $softmean$.

Annotation Provenance: Annotations are assertions made by users regarding an object. For example, a user can annotate a workflow activity or data set with a summary of its origins or perceived quality.

Dependency provenance can be used to answer the First Provenance Challenge (FPC) queries FPC #4 and FPC #6 presented above, while annotation provenance can be used to answer query FCP #8; other queries from the challenge fit into one of these two categories as well.

In our system, REDUX, provenance for an experiment is captured by a set of distinct levels, to which the result is incrementally bound. The model, in particular Levels L2 and L3, capture dependency provenance, so the question is how to best store this provenance data in the database. Provenance data from a single experiment can be stored in the following ways:

- Single provenance relation: In this method, a single relation is maintained for each level of our provenance model. As the experiment unfolds, a dependency relationship is established between separate levels and the provenance of both levels are retrieved from the database, updated and stored back into the database.

- Multiple provenance relations: In this method, multiple relations are maintained for each level, and keys are used to maintain the relationship between two relations. A new relation is added to an experiment's result provenance when, for example, a service is bound to a workflow, or an activity is executed, etc. To illustrate, we present the schema for Level L0, the abstract workflow provenance level, in Figure 3.5, Part 1:

In the first schema, abstract activities are stored in the $ActivityType$ relation. Predefined properties, such as proxy for $InvokeWebService$ are linked in the $ActivityType\_-Property$ relation (this relation links $Property$ and $ActivityType$). Each property has a $DataType$, which can be a basic data type or semantic type. The $WorkflowType$ contains basic kinds of workflows, such as sequential, state machine, event driven, etc. A tuple in the relation $AbstractWorkflow\_ActivityType$ represents a different workflow activity (of some $ActivityType$) that belongs to the $AbstractWorkflow$. This schema is capable of representing all abstract workflows (experiment design) that WinWF [8] can implement.

Figure 3.5, Part 2 illustrates the schemas that represent Level L2 (executable workflow). Level L2 is connected to Level L1 through foreign keys: $WorkflowModelId$ from the relation $ExecutableWorkflow$ and $WorkflowModel\_ActivityId$ from the relation

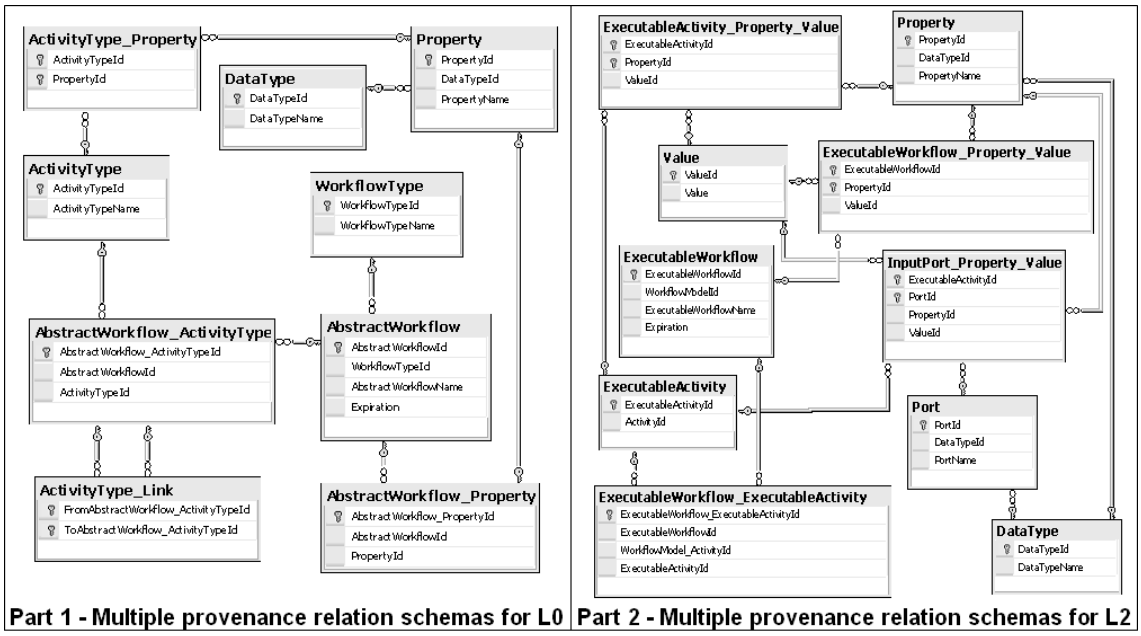


Figura 3.5: Multiple provenance relation schemas for L0 and L2

ExecutableWorkflow_ExecutableActivity. In Level L2 we record two kinds of information: i) assignment of actual values to activities and ii) assignment of input values and parameters. With this information, the workflow is ready to be executed and we can capture runtime information, such as branches taken, activities skipped or inserted at runtime, etc.

Due to space limitations, we do not present schemas for other levels in our model, and refer readers to the REDUX project entry in the *First Provenance Challenge* [43] for a detailed database design. In addition to the workflow levels, we also capture annotation provenance in our model. We permit users to attach annotation to a workflow at all levels, and they can attach annotations to other objects, such as events, individual workflow activities, parameters, input data, etc. In our current implementation, annotations are attribute name-value pairs, consisting of the name of the annotation and its value. Annotation metadata is represented with a separate table for each annotation type. For example, all annotations with Integer values are stored in one table and all attributes with String values are stored in a separate table. This approach allows indexing of annotations based on annotation type.

3.4 Related Work

This section discusses issues, advantages and possible combinations of other approaches presented in the First Provenance Challenge. A summarized figure of the approaches can be found in [76]. We highlight and extend the discussion on provenance modeling, workflow visualization, and query language and query visualization.

There is great diversity in the modeling of workflows and their provenance. But the most interesting thing to be discussed here is a common concern found in other participant approaches: information reuse. To facilitate sharing and re-usability we model our experiments in four abstraction layers. Another interesting approach is the use of *templates* presented by [57]. This facilitates the instantiation of a workflow facilitating the reuse. We can compare the *workflow template* approach with our *abstract workflow/workflow model* approach. One important difference of our system is that we represent all the layers in the same data model. Thus the user can insert details from the *executable workflow* before finishing, constructing the *abstract workflow*. The MyGrid [109] approach also allows the user to work with different granularities. Still discussing the workflow model, the Vistrail model [89] allows the storing of multiple versions of a workflow. This is a scalable solution that allows the reuse of data among the several versions of the same workflow. REDUX allow the construction of workflows as arbitrary graphs (with cycles). The majority of teams (e.g., [60, 74, 109]) represents workflows as DAGs (Directed Acyclic Graphs). This solution is fine to prevent deadlocks but restricts the construction of workflows (it does not allow the presence of loops).

To facilitate the use (and re-use) of workflow activities, one common solution is to store the full API of any activity in the provenance model [89, 109]. Our system and others (e.g., [89]) had problems to identify the interface of the activities in the workflow of the Provenance Challenge. However, this system works well with Web Services. The workflow visualization strategies vary among the solutions. Some approaches do not make a distinction (graphically) between the activities and the data in the workflow [109]. Other approaches present graphically all details about the workflow execution [57]. One interesting approach is the visualization strategy of workflow evolution presented by [89] where each node corresponds to a different version of a given workflow. Our solution allows the navigation among the abstraction layers (from executable workflow to abstract workflow and vice-versa) but does not provide an easy navigation mechanism among workflows.

Our system uses the SQL query facilities enacted with a graphical tool to facilitate queries about workflows (and their provenance). To facilitate this queries some Provenance Challenge participant teams developed their own query language [89] what can deals very well with workflow and provenance queries. Other interesting approach was

the use of query mechanisms over structured data (as XML or RDF data) - such as, the use of SPARQL. One important issue about queries that was not yet addressed by our approach is about query answer visualization.

We developed a prototype to allow the visualization of sub-workflows that are answers of a given provenance query. However, we did not develop mechanisms to facilitate the visualization of complex data objects. These mechanisms can be found in the work of [89, 109] where the user can graphically visualize the classification of the resulting objects. This approach, when combined with the use of trees to present the results of a query, seems to be a great solution.

3.5 Discussion

In this paper, we argue the collection of provenance data should be the responsibility of the workflow system, and the resulting provenance data managed by a relational store. We present a snapshot of the REDUX project. In our system, provenance collection and management are transparent, so users do not have to take any special actions to have the provenance of their experiment result collected and maintained. By making the system responsible for the generation and collection of result provenance during execution, we free the scientists from having to track provenance manually. We envisage this provenance data to be a first class data resource in its own right, allowing users to both query experiment holdings to explain or establish trust in a result, and to drive the creation of future experiments.

We present a multilayer model, to incrementally capture the provenance that a result is bound to. This model allows users to navigate from abstract levels into lower detailed execution levels, depending on the amount of detail required to validate a result. Our representation also allows a scientist to specify exactly what they wish to share, or retain, from an experiment. When stored in a relational database this provenance data can be indexed and queried as a first class data product using conventional tools and techniques. Moreover, we can apply techniques that make the storage of this data product economically feasible.

Our system handily dealt with the workflow in the Provenance Challenge though our single difficulty was to identify a detailed API of the workflow activities. Almost all approaches presented in the First Provenance Challenge adequately addressed the challenges of the event and were able to answer the provenance queries. The interchange of information among the participant groups allowed us to identify aspects that can be improved in our system and we look forward to the next generation of provenance systems that will be the result from this event.

Capítulo 4

An ontology-based framework for bioinformatics workflows[4]

4.1 Introduction

Scientific workflows [106] are being increasingly adopted as a means to specify and coordinate the execution of experiments that involve participants in distinct sites. Such workflows allow the representation and support of complex tasks that use heterogeneous data and software [22]. They differ from business workflows in several points. In particular, in bioinformatics they are characterized by a high degree of human intervention and variability in workflow design for the same task.

Bioinformatics workflows are often specified manually, and tasks are redefined from scratch (e.g., using script languages) [22]. With the advent of distributed execution of workflows (e.g., in grids [99]), task definition is sometimes being replaced by the invocation of a Web service that performs that task [46].

Manual composition is a hard work and susceptible to errors. In bioinformatics, due to the constant evolution of the area and the combinatorial explosion of tools, there are too many alternatives for workflow construction and choice of appropriate services. Thus, there is a need for means to help scientists to design appropriate workflows. Another important issue is that of traceability, to ensure the quality of an experiment.

The main idea behind our work is to take advantage of ontologies to support the specification and annotation of bioinformatics workflows, and to serve as the basis for tracking data provenance [42]. An underlying assumption is that the problem of automatic or iterative composition of workflow tasks can be seen as an Artificial Intelligence (AI)

[4]**L.A. Digiampietri, J.J. Pérez-Alcázar, and C.B. Medeiros. An ontology-based framework for bioinformatics workflows.** ***International Journal of Bioinformatics Research and Applications*****, 3(3):268–285 2007.**

planning problem [65]. We extend AI planning techniques with ontologies to create a semantic framework for design, reuse and annotation of bioinformatics experiments.

The paper attacks the problem of constructing and annotating scientific workflows, under the assumption that they are the basis for specifying and executing tasks in a distributed (laboratory) environment. Each activity within such a workflow can be executed either by invocation of a Web service or of another (sub) workflow.

The paper's main contributions are thus:

- proposing a solution to the problem of composition of services, combining results from AI and ontology management, thereby helping design scientific workflows, while at the same time documenting design alternatives;
- using ontology repositories to enhance the semantics in automatic workflow construction and facilitate tracking data and procedure provenance;
- validating the proposal by means of a prototype for genome assembly and annotation.

Our implementation takes advantage of WOODSS (WOrk-flOw-based spatial Decision Support System) [71], a scientific workflow infrastructure. Originally conceived for decision support in environmental planning, it has evolved into an extensible database-centered environment that supports specification, reuse and annotation of scientific workflows and their components.

The rest of the paper is organized as follows. Section 4.2 describes related work. Section 4.3 presents our architecture. Section 4.4 discusses our prototype. Section 4.5 contains conclusions and ongoing work.

4.2 Related work and concepts

4.2.1 Workflows and Web services in bioinformatics

The *genome assembly* problem consists in joining and matching together pieces of DNA sequences to create a cogent sequence, much in the way crossword puzzles are put together. Constituent sequences are created inside a laboratory by procedures that extract pieces from a species' DNA and then produce long strings of base pairs (ACGT). Challenges in this process include the adequate generation and annotation of sequences, as well as finding the appropriate means of assembling them together into an accepted genome.

Genome annotation is the assignment of functions to each gene. The empiric verification of gene functions is a time- and money-consuming activity. Most functions are assigned based in similarity between the DNA sequence of the target gene and the sequences

of already annotated genes. Gene annotation can therefore be partially automated, but manual data verification is always recommended.

Genome assembly and annotation are composed by several complex activities, involving interactions among basic tasks, human intervention and access to heterogeneous data sources. A trend in the bioinformatics community is to see each such experiment as a workflow designed by scientists to help their daily activities [82]. However, this practice has little flexibility, hampering the edition and reuse of these workflows.

Several other problems are involved in the construction of such workflows. Among them, this paper is concerned with: (1) data and software tool provenance; (2) tool/task composition, translated into a problem of Web service composition; and (3) mechanisms for finding the appropriate tools or services to execute a task. We use domain ontologies as the basis for attacking these problems. The first question - provenance of data and software tools - directly affects the acceptance of the results of experiments. The quality of bioinformatics experiments depends on properly identifying data origins and the processes that produced these data [17]. At most times, provenance is indicated by laborious manual annotations, which often vary across laboratories.

The second issue concerns tool/task composition while constructing the workflows[22, 108, 71]. We highlight three kinds of composition: manual (supervised), iterative (using top-down design practices) and automatic. In a scenario where several software tools are being made available on the Web, the composition problem has become more important. To help this issue, many tools invoked by such workflows are now encapsulated into Web services.

Our third problem is how to find tools on the Web that execute some task. This search is typically based on keyword queries [17] that can however return several unwanted results and may not find the desired tools. Even when found, the integration of a tool with the user's system is not easy. Thus laboratories rebuild tools, replicating work and decreasing tool sharing and reuse.

The Semantic Web has been proposed to solve interoperability and discovery problems. However, this will require extending the languages to add semantics to service description and discovery - e.g., using ontologies. The development of Semantic Web services must address the challenges of automatic discovery, invocation, composition and monitoring of service execution.

4.2.2 Planning and Composition of Web services

Automatic composition of Web services is a recent trend to meet some of the challenges and problems mentioned in the previous section. Users (in this paper, bioinformatics scientists) should be able to specify "what" they desire from the composition (high level

goals and actions), and the system supplies the "how" - the Web services to be used, how to interact with those services, etc. The process of composing the services must be transparent to the users, and the detailed descriptions of the composed services must be generated automatically by the system from the users' specifications.

Among the proposed solutions for the automatic composition problem we mention those based on planning, exploited by us, and those based on workflows. Workflow-based composition methods can be divided into static and dynamic workflow generation [86]. In the first case, the workflow is specified manually and only the selection and binding of atomic Web services with the workflow is automatic. Dynamic composition automatically creates the workflow and selects atomic services, e.g., [28, 45].

The task of presenting a sequence of actions to achieve an objective is called in Artificial Intelligence *plan synthesis*, or *planning* [47, 87]. Planning is a mature area in AI, with well-studied algorithms. Such techniques are currently used in mobile robots, manufacturing processes, emergency management, among others [78, 81].

Recent research efforts have investigated the use of planning to solve the problem of automatic composition of Web services [54]. According to [95], in order to use planning in this context, AI planning concepts must be extended to consider factors such as complex control structures with loops, non-determinism and conditionals.

We highlight other important characteristics, not usually found in AI planning, such as: the use of non-functional attributes, like cost or quality, which can facilitate the choice of the plan most adequate to the user's requirements; the need to support semantic constructions such as hierarchies (abstractions), as well as compatibility with the different Semantic Web service description standards, like OWL-S (www.daml.org/services) and WSMO (www.wsmo.org); and the support of extended goals involving complex conditions on process behavior.

Many planning systems and algorithms have been considered as candidates for supporting composition of Web services. For instance, the work of [70] extends Golog [64], a language based on situation calculus, to allow automatic building of Web services. Another option is the use of PDDL (Planning Domain Definition Language), a widely used formal language, whose notation is similar to OWL-S and thus a good candidate to use in specifying Web services composition. Other solutions involve rule based planning methods [72], and symbolic model checking, which has also been used to automatically compose services described in OWL-S [102].

Yet another method is *hierarchical planning*, to support iterative and automatic composition of services to specify workflows. Hierarchical planning is an AI planning methodology that creates plans by task decomposition. One well-known hierarchical planner is SHOP2 (Simple Hierarchical Ordered Planner 2) [80] which is based on Hierarchical Task Network (HTN) [87]. SHOP2 won the prize of one of the four best planners in the 2002

International Planning Competition [65].

Sirin et al [93] use SHOP2 for the automatic composition of Web services, and the inputs to their planner are specified in OWL-S. The authors claim that automatic task decomposition using HTN planning is very similar to the concept of complex process decomposition used in OWL-S ontologies.

None of the planning proposals mentioned treats complex objects or objects created dynamically, two very important characteristics within Web services. Moreover, planning algorithms do not consider the existence of relationships among objects which might result in plan improvement. As will be seen, we solve these issues by extending SHOP2 to take advantage of ontology repositories.

4.3 An architecture for automatic composition via planning

This section presents our solution to the problem of helping scientists design scientific workflows. As will be seen, one important issue is the possibility of reusing parts of workflows constructed by other scientists. Another relevant issue is annotation. Reuse and annotation are supported by ontologies, which also guide the planning algorithm.

Our plans are specifications of scientific workflows. Thus, in this section, we will use indistinctly both terms.

4.3.1 Repositories

Our workflow design and execution process is based on combining AI planning techniques with information stored in three repositories: *Ontology Repository*, *Service Catalog* and *Workflow Repository*. While the Ontology Repository contains information about domains and service types, the Service Catalog stores information about service instances.

In more detail, the *Ontology Repository* contains two kinds of ontology (Domain and Service) that are used to support automatic composition and annotation of services and workflows – in our case study, information about genome assembly and annotation, see Section 4.4. The concepts in the Domain Ontology describe a given application domain. The concepts in the Service Ontology describe the different kinds of services and their relationships. This ontology is used in automatic composition to help check compatibility among composed tasks (e.g., interface matching). The Service Ontology does not store descriptions of the services themselves. Rather, it contains what we choose to call "service type" - i.e., a generic description of each kind of service, its generic interface, parameters, etc. Thus, it will contain a description of "alignment" services, but no instantiation of this type of service. Service instantiation is left to the Service Catalog.

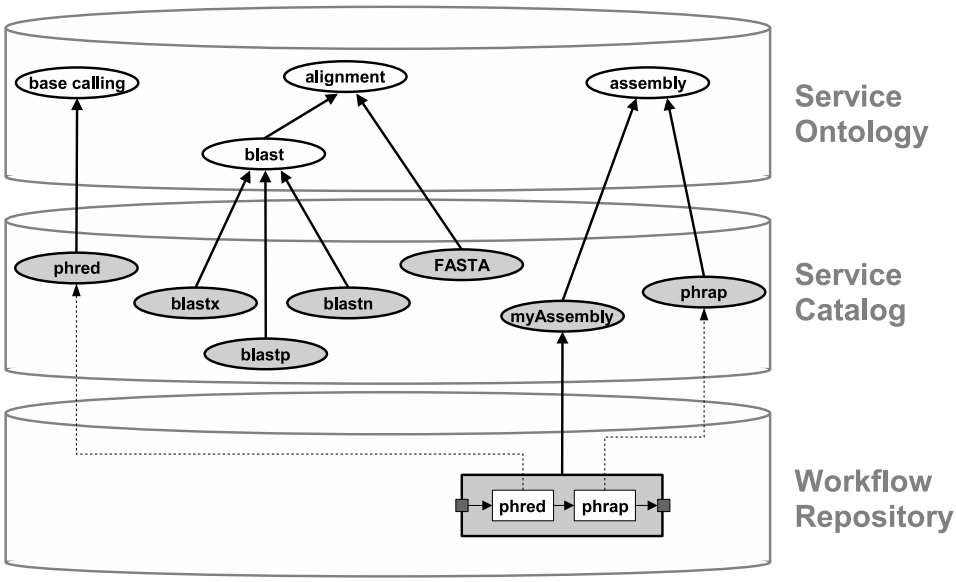


Figura 4.1: Parts of our service ontology. Service types appear in white ovals, service instances in dark ovals

Figure 4.1 shows the "is a" relations of some services in the Service Ontology (service types) and the services in the Service Catalog (service instances) - e.g., a *FASTA* service "is a" service of the type *alignment*. Our repositories also store "aggregation" relations, for example, the service *myAssembly*, in the Service Catalog, can be associated with a workflow, in the Workflow Repository, that is an aggregation of the services *phred* and *phrap*. This kind of relationship is used in our iterative composition process (in each iteration, the system suggests to the user how to decompose an abstract activity into more concrete activities).

The *Service Catalog* plays the role of a UDDI (Universal Description Discovery & Integration) registry, enhanced with extended functionalities. Standard UDDI structures store information about service providers, the Web services they make available, and the technical interfaces which may be used to access those services ports. Our Service Catalog, besides, stores a service's non-functional attributes (metadata), such as execution time, quality, reliability and availability, used to rank workflows. Figure 4.2 presents the standard UDDI Web services register approach and our extension to this approach. Each service entry in the Catalog is annotated with the ontological concepts of the Service Ontology. The port types are annotated with concepts from the Domain Ontology.

The *Workflow Repository*, adopted from WOODSS [71], stores annotated (sub) workflows at different abstraction levels, from abstract specifications to runtime workflows. The Workflow Repository also stores all annotations and information on data needed to

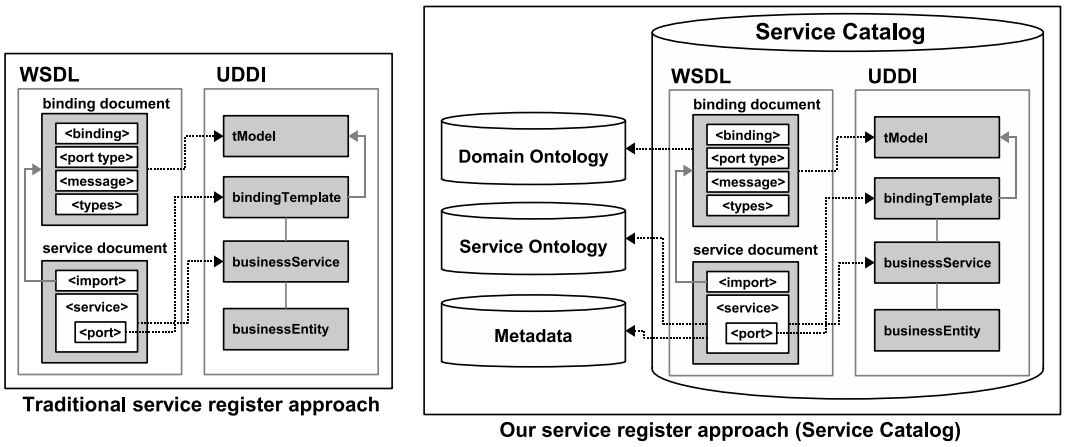


Figura 4.2: Service Register approach: UDDI approach and our semantic approach

run a given executable workflow. This includes pointers to files that store intermediate execution results and metadata associated with each execution (e.g., timestamps, actors involved).

The repositories are interrelated as follows. Workflows and data in the Workflow Repository are annotated with terms from both ontologies, the services invoked within concrete workflows come from the Service Catalog. Moreover, concepts of the Service Ontology are used to annotate terms in the Service Catalog; and concepts in the Domain Ontology annotate types stored in the Service Ontology. Section 4.4.1 discusses these interrelationships for our case study.

4.3.2 The composition architecture

Our architecture is able to deal with automatic composition of workflows based in Web services. Figure 4.3 shows this architecture, highlighting the main modules and their interactions. Only the main connections among modules are represented. Section 4.4 presents our implementation of this architecture.

The Interface Layer allows the user to design, search, edit, execute a workflow and store it in the repository. It also allows a user to register services and workflows, request the execution of a workflow and interact with this execution. Furthermore, it supports syntax verification and suggestions of activities; automatic specification through AI planning; and iterative composition.

The Service/Workflow Discovery module is responsible for the search of services and workflows that meet user requests. Search can be based on context, syntax and functionality. Search for context is based on ontological annotations of services. Search on syntactic compatibility is based on the parameters of the services' interfaces. Search for

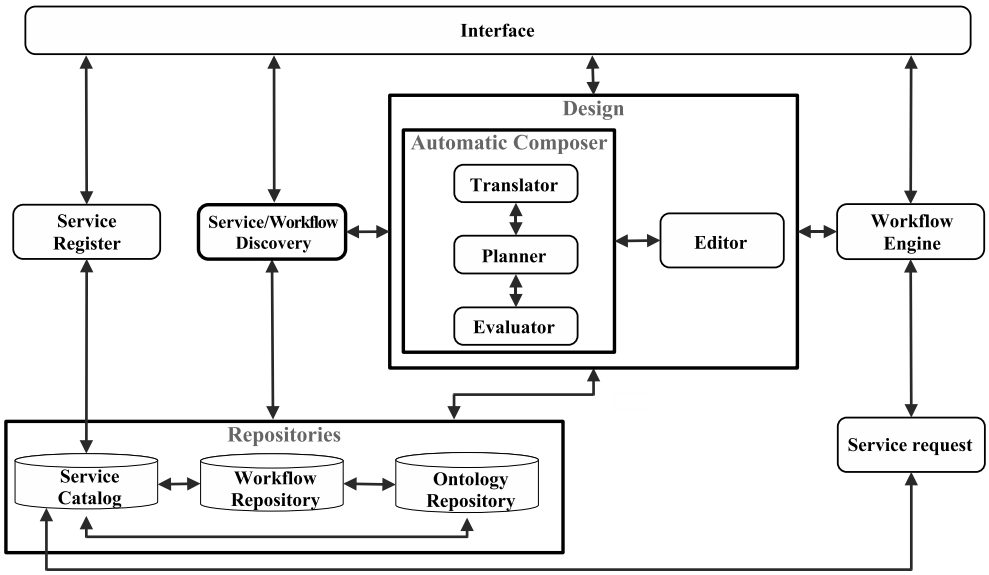


Figura 4.3: System Architecture

functionality is based on keywords, and can be local (Repositories) and global (on the Web). Whenever the global search returns a service that is not already registered in the Service Catalog, the user is required to validate and register the appropriate answers in the local Repository. Hence, the Repository contains only properly annotated services, whose provenance and quality are guaranteed by some expert. When no service or workflow meets the requests, this module will ask the Design module to create new workflows.

The Design module is responsible for constructing a workflow, which at any time can be edited by a scientist (the Editor box). The Automatic Composer encapsulates the Translator-Planner-Evaluator modules of AI planning, see [86]. It receives a plan request R from the interface and generates workflows automatically or iteratively. To generate these workflows, the Translator needs first to convert the request to the planner's language. Next, the Planner interacts with the Workflow and the Ontology Repositories to obtain information for plan generation, and with the Service/Workflow Discovery facility to check for existing available services.

Rather than generating executable workflows, our planner produces abstract workflow specifications. The reason is that plans refer to *service types* (defined in the Ontology Repository), rather than to the services themselves (whose specification is stored in the Service Catalog). This choice was made mainly to improve efficiency and scalability in the planner [3]. The Evaluator converts these abstract workflows into concrete ones and chooses among them the workflow that best suits the request R. This selection is based on the non-functional attributes (execution time, quality, reliability, etc) that annotate services in the Catalog and can be guided by the user. The workflow can then be forwarded

to the Workflow Engine, where the user can provide the input data and start its execution.

The Editor module supports workflow design. It accesses the workflow repository and lets the user manually compose, reuse and annotate workflows. Annotations include free text and references to the ontology repository. This repository can also be updated (e.g., adding or modifying an ontology). However, this is outside the scope of the architecture and is left to specialized tools - e.g., Protégé [59] - since the architecture's goal is not to manipulate ontologies, but to use them to help experiment annotation, reuse and specification.

The user interacts with the Service Register module in order to define new services. These services are described in WSDL and OWL-S, and linked to the Ontology Repository. These services can be those developed and available locally, or those that are available elsewhere, but whose provenance has been certified by the user. New service types are registered in the Ontology Repository using an ontology editing tool. New services are entered into the Service Catalog using the Service Register Module.

4.3.3 Execution environment

The Workflow Engine follows the specification of the Workflow Management Coalition (http://www.wfmc.org). It is responsible for workflow execution and supports user interaction, e.g., to validate or interrupt execution flow. It is responsible for controlling the execution of all workflow activities. The operations provided by the Workflow Engine are: interpretation of the complex process definitions; creation and management executable workflows; and supervisory and management functions. It sends the requests (and parameters) for service invocation to Service Request.

The Service Request module is responsible for the management of each Web service request, communicating with the Web server provider, sending input data and receiving the results. This module also detects service faults. Faults are registered and used to calculate service non-functional attributes (such as availability).

There are three alternatives to solve a fault. The first is try to re-execute the service that presented the fault. This is useful when a service is unavailable during a short period of time. The second is to replace the faulty service by an equivalent service (of the same type but with less ranking, following non-functional attributes, in our Service Catalog). In this case, the Workflow Engine module can ask the Evaluator module for the selection of an alternative equivalent service to replace the faulty service. If these alternatives do not work, the Workflow Engine can request new plans to solve the problem.

This architecture supports the three kinds of composition presented in Section 4.2.1. In manual composition, the system will only let a user combine two activities if their inputs and outputs are ontologically compatible. Ontological compatibility is based on

subsumption properties, see [42]. In iterative composition, in each iteration the system suggests to the user tasks or sub-workflows that have been previously stored in the repository and that can be used for an already defined task. In automatic composition, it designs a set of workflows that satisfy the user's requests.

4.3.4 User interaction

Users can interact with the architecture to annotate and design workflows, monitor and change their execution and register services. One of the most important user interactions is the request for a plan (i.e., the construction of a new workflow to meet a specific request). This process starts when a user (human or software agent) makes a request for a service.

This request can be the description of a goal or task. Starting from it, the Planner generates alternative plans to meet the request. In this process, the planner accesses the Domain and Service ontologies to obtain the necessary information for the planning process. Once the plans are generated, they are passed on to the Evaluator, which chooses the best plan to meet user needs.

Domain and Service ontologies, stored in the ontology bases, are key concepts to this process. Initially, they are used by the Translator to generate a request to the planner, disambiguating the user's demand for a service. Next, these bases are used by the Planner to generate the appropriate service compositions. The Planner accesses them to obtain the functionalities of the services and generates an abstract scientific workflow. The Planner uses the domain ontology to improve the efficiency of the planning process and to facilitate the modeling and the management of complex objects. The planner's output contains several workflows (the plans) with equivalent or similar functionalities.

4.4 Case study: genome assembly and annotation

We implemented a prototype of the architecture presented in Section 3 to solve the problems of genome assembly and annotation. We decided to use SHOP2 in our implementation because it provides the following benefits: (a) it supports embedding domain knowledge to control the search space and improve efficiency; (b) it has been successfully used in a variety of real-world planning-based applications; (c) it allows inclusion of different types of pre-condition constraints for service operators as well as calls to external systems; (d) it enables reuse by facilitating selection of appropriate methods from domain-related operator libraries. Section 4.4.2 shows how we extended the SHOP2 language to support complex objects and enhanced semantics.

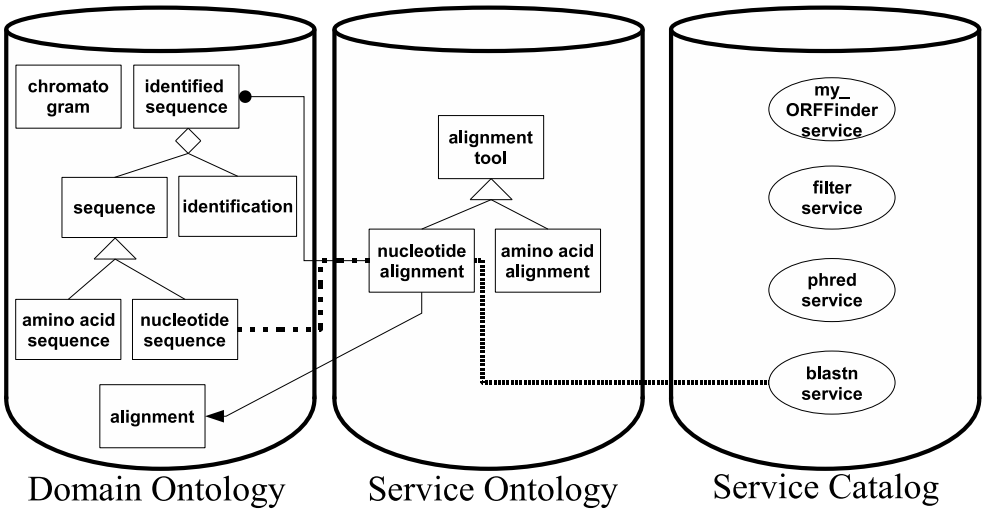


Figura 4.4: Small example of the relationships among our repositories

4.4.1 Repository instantiation

In order to implement the architecture, we had to construct the appropriate ontologies. Ontology editing uses Protégé [59], a well known ontology editing tool.

Several bioinformatics ontologies have been proposed [94, 98]. Some are very detailed taxonomic ontologies, used to cluster [107] and/or to annotate data [94]. Others are concerned with describing relationships among bioinformatics tools (akin to our service ontology) or objects manipulated in bioinformatics (e.g., our domain ontology). The latter are still under development, and are usually defined independently. This hampers their use in an integrated manner. Ontology management techniques, such as alignment or integration [56] are not suitable for this kind of need, since they apply when ontologies cover associated concepts.

Since our goal was to help scientists to specify and manage their experiments, we developed detailed domain and service ontologies, specific to genome assembly and annotation, extending a generic bioinformatics ontology [98] and specifying relationships between these two ontologies. As a consequence, they can be considered to form a (single) complex ontology covering services and domain aspects.

Using our ontology repository, we annotated bioinformatics data and tools in order to allow semantic search and automatic composition of services. Figure 4.4 shows a small portion of our ontologies and their interrelationships. Domain and Service portions are separated, thus helping establish distinct relationships among the concepts. In particular, the figure shows how concepts in the Service Ontology help qualify services, and how concepts in the Domain Ontology help define service parameters. To simplify the figure, we omitted several relationships.

We highlight only the relationships (links among the repositories) involved with *nucleotide alignment*. The *blastn service* entry in the Service Catalog corresponds to a service that implements a *nucleotide alignment*, which is an *alignment tool* description (Domain Ontology annotation). The *nucleotide alignment* has as input an *identified sequence* and as output an *alignment*, both concepts of the application domain, duly annotated by appropriate references in the ontology. The dotted line between *nucleotide alignment* and *nucleotide sequence* indicates a restriction on input data: the *identified sequence* input to the *blastn service* must be a *nucleotide sequence*.

In our Domain Ontology, all concepts are atomic data types (integer, strings, etc) or, recursively, an aggregation and/or a generalization (or specialization) of concepts. We can observe in Figure 4.4 that *nucleotide alignment* is a specialization of *alignment tool* and *identified sequence* is an aggregation of *sequence* and *identification*.

The Domain Ontology also provides semantic annotation for workflows and data. This is exemplified in Section 4.4.3.

4.4.2 Planning with ontologies

A SHOP2 specification [80] is composed of three sections: domain definition (*defdomain*); problem definition, defining the problem that the planner must solve (*defproblem*); and requests to find the plans that solve a given problem (*find-plans*).

SHOP2 domain definition requires *methods* and *operators*. Operators specify available activities to implement methods. Methods are used in the planning process – a plan, at its highest level, is a concatenation of method invocations, recursively refined into methods and operators. Methods provide a convenient way to write problem–solving "recipes" that are used by SHOP2 in order to solve problems, they correspond to activity types in our Service Ontology or workflows in the Workflow Repository. Figure 4.5 shows part of the translation of our Bioinformatics Domain and Service Ontologies into SHOP2. This translation is automatic, and performed by the Translator module.

From a high level point of view, our translator produces a *defdomain* specification enriched with concepts from the Ontology Repository. Concepts in the Service Ontology and relationships in the Domain Ontology are translated into operators. Service and Domain concepts are subsequently translated into methods. Translation works as follows. We now explain in detail how ontologies are introduced to help planning (i.e., workflow construction).

Each "type of service" concept in our Bioinformatics Service Ontology generates a SHOP2 operator whose pre-conditions are the service's input data types and whose post-conditions are the output data types. For example, lines 2, 3, 6 and 7 of Figure 4.5, respectively, show the service types *ORFFinder*, *nucleotide alignment*, *filter* and *base*

```
1  (defdomain bioinformatics-ontology (
2    (:operator (!orfFinder ?a) (sequence ?a) () ((ORFS ?a)) 25.0)
3    (:operator (!nucleotide_alignment ?a) ((identified_sequence ?a))
()((nucleotide_alignment ?a)) 50.0)
4    (:operator (!decompose_identified_sequence ?a) ((identified_sequence ?a)) ()
((nucleotide_sequence ?a) (identification ?a)) 0.0)
5    (:operator (!compose_identified_sequence ?a) ((nucleotide_sequence ?a)
(identification ?a) (not(identified_sequence ?a))) () ((identified_sequence ?a)) 0.0)
6    (:operator (!filter ?a) ((sequence ?a) (not(filtered ?a))) () ((filtered ?a)) 4.0)
7    (:operator (!base_calling ?a) ((chromatogram ?a))()((identified_sequence ?a)) 10.0)
8
9    (:method (TO_identified_sequence ?x)
10     ((chromatogram ?x))
11     ((!base_calling ?x))
12     ((nucleotide_sequence ?x)(identification ?x))
13     ((compose_identified_sequence ?x))
14   )
15
16   (:method (find_orfs ?x)
17     ((nucleotide_sequence ?x))
18     ((!orfFinder ?x))
19   )
20
21   (:method (TO_alignment ?x)
22     ()
23     ((TO_nucleotide_alignment ?x))
24     ()
25     ((TO_aminoacid_alignment ?x))
26   )
27
28   (:method (TO_nucleotide_alignment ?x)
29     ((identified_sequence ?x))
30     ((!nucleotide_alignment ?x))
31     ( not (nucleotide_sequence))
32     ((:ordered((TO_identified_sequence ?x) (!nucleotide_alignment ?x))))
33   )
34
35   (:method (TO_aminoacid_alignment ?x)
36     ((aminoacid_sequence ?x))
37     ((!aminoacid_alignment ?x))
38     ( not (aminoacid_sequence))
39     ((:ordered((TO_aminoacid_sequence ?x) (!aminoacid_alignment ?x))))
40   )
41  )
42 )
```

Figura 4.5: Part of the SHOP2 definition domain for bioinformatics ontology

calling translated into SHOP2 operators. In more detail, *nucleotide alignment* is a type of service described in SHOP2 (line 3, Figure 4.5) as an operator whose pre-condition is the existence of an *identified sequence* (input) and whose post-condition is the production of a nucleotide alignment (output). The value of 50.0 for this line indicates that the user provided this value as an upper bound estimate for the execution time of any *nucleotide alignment* service. As will be seen later on, the Evaluator will suggest instantiations for each service type, and use these values to rank suggestions.

A SHOP2 planner does not support generalization/specialization hierarchies of operators (tasks), neither does it manipulate complex objects. Thus, we had to extend this planner to fully support both facilities. To allow complex objects, the Translator creates SHOP2 operators that are used to represent ontological relationships. Complex objects are structures created from basic objects (basic concepts in our Domain Ontology). Since our ontologies contain aggregation and specialization relationships, we use this knowledge to improve SHOP2 functionalities. For each aggregation relationship, we created an operation called *Compose*, with cost zero. Its input specifies the concepts that will be aggregated and its output is the aggregated concept (e.g. in line 5 of Figure 4.5, an *identified sequence* aggregates a *nucleotide sequence* and its *identification*). Similarly, we created the converse *Decompose* (e.g. line 4 of Figure 4.5). To represent specializations/generalizations, for each concept that is a specialization, we create an operation called *IsSpecializationOf* with cost zero that, given the specialized concept, returns the corresponding general concept. Similarly, we created the converse *IsGeneralizationOf*. These four operators extend SHOP2's planning capabilities. Whereas the other operators are transformed into service invocations by the Evaluator, these four serve only to help compose complex tasks in a plan.

Once all operators have been specified, methods can be defined. For each type of service in the Service Ontology, the Translator creates a SHOP2 method that describes which operator (or set of operators) can execute this method. For example, lines 21, 28 and 35 of Figure 4.5, respectively, show the headers of the methods *alignment*, *nucleotide alignment* and *aminoacid alignment*.

In bioinformatics, many times, the user needs to obtain a certain result without knowing what tasks produce this result (or concept) – for example, starting from a chromatogram (input concept), he/she wants to obtain an alignment (output concept). To allow the user to pose such requests, the Translator needs to create additional methods whose execution will generate concepts in the Domain Ontology.

4.4.3 Creating and instantiating plans

The SHOP2 *problem definition section* (see Figure 4.6A) is composed by a problem name (e.g. problem1), the label of the definition domain (e.g. bioinformatics ontology), the state of the world (a set of conditions that are true in a given instant) and the set of methods that must be utilized by the planner (e.g. *nucleotide alignment ch1*). The SHOP2 *request to find plans* (see Figure 4.6B) identifies the problem to be solved and requirements for the solution plan (such as maximum cost).

Plan specification

We now clarify the process of plan specification using an example. The user requests a plan to transform a *chromatogram* (input concept) into a *nucleotide alignment* (desired concept). Part A of Figure 4.6 shows the input request (already translated to SHOP2), indicating that the user wants the system to produce a workflow for a *nucleotide alignment* using *chromatogram ch1*. Notice that the user defines the request, but does not need to specify how to accomplish it. Part C of this figure shows the four possible plans (abstract workflows) generated by SHOP2 to answer the request, ordered by increasing cost order.

Plans are sets of methods that must be executed sequentially to achieve the goal. For instance, the first plan *((!BASE CALLING CH1) (!NUCLEOTIDE ALIGNMENT CH1))* shows that a possible solution is to execute a service of type base calling, using *ch1* as input, and passing the result of this execution to a service of type *nucleotide alignment*. Symbolic variable *ch1* is a *chromatogram* as defined in the *defproblem* section - Figure 4.6A.

This figure also shows, for instance plans 3 and 4, the need for ontological concept manipulation. These plans chose the *ORFFinder* service type, which needs the *sequence* concept as input (see line 2 of Figure 4.5). However, the problem's input is a complex data object of type *identified sequence* (see Figure 4.4). Thus, in order to feed it to *ORFFinder*, the planner had to use our *Decompose_identified_sequence* and *Compose_identified_sequence* ontological relationship operators to, respectively, disaggregate the problem's input and re-aggregate the output from *ORFFinder*.

Plan instantiation

The four plans generated are abstract. For instance, a *nucleotide alignment* can be implemented by services *blastn* or by *FASTA*. Abstract plans are forwarded to the Evaluator that will generate an executable workflow by finding the Web services that best fit each abstract service in the plan, using the ontological annotations of the Service Catalog. Our Evaluator takes several criteria (metadata from the Service Catalog) into consideration to select the most appropriate plan. These characteristics are retrieved by the Evaluator

A. SHOP2 input problem

```
(defproblem problem1 bioinformatics-ontology
  ((chromatogram ch1))
  ((nucleotide_alignment ch1)))
```

B. SHOP2 request to find plans

```
(find-plans 'problem1 :verbose :plans :which :all :optimize-cost 200)
```

C. SHOP2 output

```
Problem PROBLEM1 with :WHICH = :ALL, :VERBOSE = :PLANS, OPTIMIZE-COST = 200
Totals: Plans Mincost Maxcost Expansions Inferences  CPU time  Real time
          3    60.0    85.0        29        70     0.017      0.042

Plans:
(
 ((!BASE_CALLING CH1) (!NUCLEOTIDE_ALIGNMENT CH1))

 ((!BASE_CALLING CH1) (!FILTER CH1) (!NUCLEOTIDE_ALIGNMENT CH1))

 ((!BASE_CALLING CH1) (!DECOMPOSE_IDENTIFIED_SEQUENCE CH1) (!ORFFINDER CH1)
        (!COMPOSE_IDENTIFIED_SEQUENCE CH1) (!NUCLEOTIDE_ALIGNMENT CH1))

 ((!BASE_CALLING CH1) (!FILTER CH1) (!DECOMPOSE_IDENTIFIED_SEQUENCE CH1)
        (!ORFFINDER CH1) (!COMPOSE_IDENTIFIED_SEQUENCE CH1)
             (!NUCLEOTIDE_ALIGNMENT CH1))
)
```

D. Workflow – graphical representation of the chosen plan

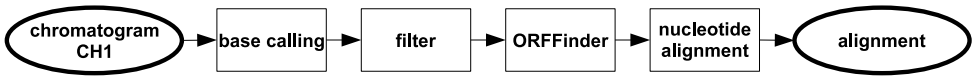


Figura 4.6: Plan synthesis and selection

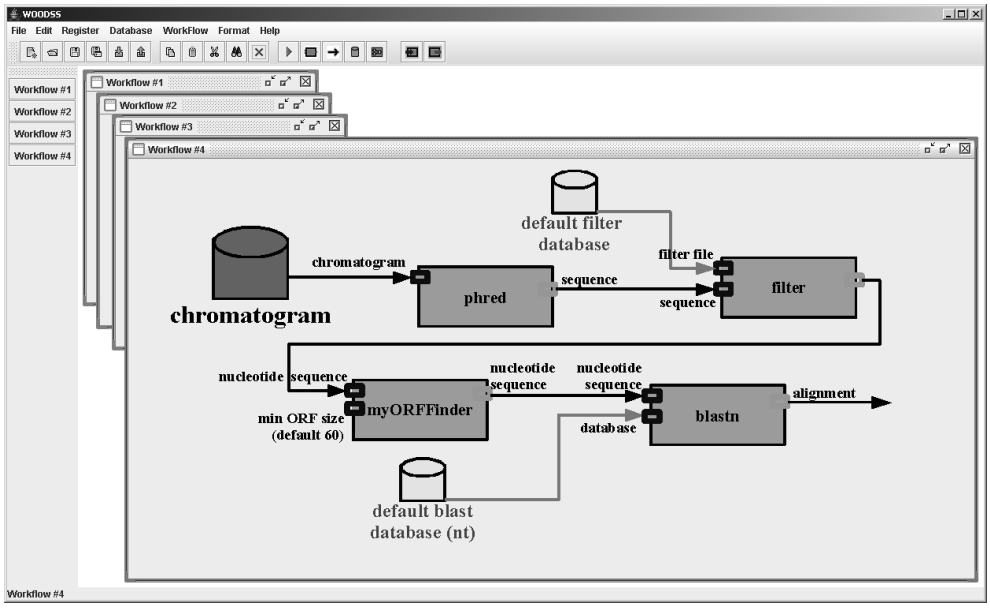


Figura 4.7: Workflow Graphical Representation adopted from WOODSS

from the Service Catalog, together with service description. Figure 4.6D shows that the Evaluator chose the fourth plan. This plan was chosen because attributes in the service Catalog state that the use of the *filter service* increases the quality of the plan results and, if the user is looking for genes, the use of the *ORFFinder service* will improve the result.

Part D of Figure 4.6 shows a graphical version of the workflow for the selected abstract plan. Figure 4.7 shows a concrete (executable) version of this abstract workflow, using our graphical interface. Activities are rectangles, transitions are arrows and data repositories are represented as cylinders. Through this interface, the user can create, edit, annotate and execute workflows.

In manual editing, to insert a new activity in a workflow, the user must select one activity from the list of available activities from the Service Catalog. The system checks the consistency of all transitions (linking outputs of one activity to the inputs of another activity). Inputs can have default values (e.g. "60" is the default value for *minimum ORF size* of the *myORFFinder* activity).

To insert new data, the user must select a data type from the list of available concepts of the Domain Ontology. Other operations allowed by the graphical interface are: embed a (sub) workflow inside a workflow, save, load, export and import workflows. This allows scientists to reuse (parts of) stored workflows to create new experiments. The user can also monitor workflow execution. For instance, he/she can click in the arrows (transitions) to verify the produced data and the metadata associated (such as the name and version

of the tool that produced this data; the date in which the data was produced; the origin of the data; etc).

Scientists can designate which annotated workflows can be stored in the repository, to be used in subsequent experiments or to be shared with other scientists. When an experiment is saved, the data produced in each step of workflow execution is also saved. Moreover, a set of annotations are assigned to each produced data set, taking advantage of ontology type and service definitions. These annotations describe the origins of the data and the service (or set of services) that produced that data, facilitating data and service provenance traceability. In the example of Figure 4.7, the user, when consulting the result alignment, sees that the alignment was produced by *blastn service*, using the *blast database nt* and, as input sequence, the *ORF* produced by *myORFFinder service*.

4.5 Conclusions and ongoing work

This paper presented an ontology-based framework to support bioinformatics work. It helps the user in the three kinds of composition: manual, iterative and automatic. It takes advantage of AI planning techniques, combined with ontologies and Semantic Web standards. The solution is based on repositories that store information on services and their characteristics, on service and domain ontologies, and on workflows. In particular, ontology repositories are extensively used in enhancing plan generation with semantics and in helping users design better scientific workflows. Several bioinformatics laboratories have reported the use of scientific workflows and of a workflow infrastructure to support their experiments - e.g.,[82]. Our work extends these approaches in three main directions: first, its use of AI planning techniques to help design the "best" workflow for a task; second, the use of ontologies to semantically support workflow construction, both in selecting tasks and in finding appropriate services; third, the use of these ontologies in annotation and thus help traceability. We have built a prototype to verify and validate our proposal, for bioinformatics problems, specifically for genome assembly and annotation.

Ongoing work concerns mechanisms for improving provenance and traceability support. We also intend to explore other extensions for plan synthesis - e.g., re-planning and plan repair. We also intend to extend our bioinformatics ontology to reach a wider context in bioinformatics, such as comparative genomics and metabolic pathways.

Capítulo 5

Traceability Mechanisms for Bioinformatics Scientific Workflows [5]

5.1 Introduction

This paper is concerned with combining research on databases and on scientific workflows applied to the development of a framework that seamlessly supports the backwards and forward traceability of bioinformatics experiments, at distinct abstraction levels. The term *traceability* is used here to denote the ability to describe the way in which some product has been developed, including all processes it went through. *Provenance*, on the other hand, is associated with a place or origin, and is usually linked to data – i.e., when, where and who produced it. Provenance normally does not demand a detailed process trace. Both issues – traceability of processes and provenance of data – are very important in any kind of scientific experiment.

In bioinformatics, for instance, the quality of an experiment depends heavily on properly identifying both data origins and the processes that produced these data [17]. However, it is common practice that provenance is captured by laborious manual annotations, which often vary across laboratories; moreover, the majority of computer systems in a laboratory do not provide traceability mechanisms. Our work concerns traceability, and thus the ability to "discover the cause or origin of something by examining the way in which it has developed" [18].

Different groups of scientists have distinct needs in terms of provenance and traceability mechanisms [10, 11]. To cope with this issue, we propose a layered and flexible solution to deal with the range of user needs. Furthermore, a bioinformatics framework

[5] **L.A. Digiampietri, C.B. Medeiros, J.C. Setubal, and R.S. Barga. Traceability Mechanisms for Bioinformatics Scientific Workflows. In *Proceedings of the AAAI2007's Workshop on Semantic e-Science (SeS07)*, 2007.**

must contain integration mechanisms to provide a transparent access to heterogeneous data and to provide interoperability among tools and systems.

Thus, the paper is concerned with how to record detailed information about all processes involved in an experiment to produce a given result, such as an assembled genome or a phylogenetic tree. Our solution is based on combining annotated scientific workflows with database mechanisms. Annotations and workflows are stored in a database, which allows supporting traceability requirements through database queries, allowing for both forward and backward tracking within a bioinformatics experiment. We are testing the framework for bioinformatics problems in the tasks of genome assembly.

The main contributions of this paper include the following: (1) Description of a framework to support the tracing of all steps of a bioinformatics experiment, taking advantage of standard database storage and querying mechanisms; (2) Extension of database query facilities to support domain-specific concerns (e.g., in the bioinformatics context, allowing queries concerning "alignment", "base-calling", etc); (3) Validation of the proposed approach via a prototype, tested using real laboratory data.

The remainder of this paper is organized as follows. Section Main Concepts and Related Work describes basic concepts and related work. Section Architecture presents our architecture. Section Implementation Issues presents examples where our system is helping scientists. Section Conclusions contains our conclusions and plans for future work.

5.2 Main Concepts and Related Work

This section discusses concepts used in the paper and reviews related work, focusing on workflow and provenance representation, data integration and interoperability, and traceability. Finally, we describe our previous work whose contributions include mechanisms to facilitate bioinformatics workflow composition [37, 36], and efficient storage of provenance information [10, 11].

5.2.1 Basic Concepts

The *genome assembly* problem consists in joining and matching together pieces of DNA sequences to create a contiguous sequence, much in the way jigsaw puzzles are put together. Fragment sequences are created inside a laboratory by procedures that extract pieces from a species' DNA and then produce long strings of base pairs (ACGT). The main challenge in this process is finding the appropriate means of assembling the fragments together into a biologically correct sequence. For more details on bioinformatics problems see [91].

One of the challenges in genome assembly is how to combine the several complex

activities that are used in these processes. These activities use heterogeneous data from distinct sites and need a high degree of human intervention. *Scientific workflows* are being used as a means to design and execute these complex activities [82]. A *scientific workflow* [106] is the specification of a process that describes a scientific experiment. Such workflows allow the representation and support of complex tasks that use heterogeneous data and software [22]. In particular, in bioinformatics they are characterized by variability in workflow design for the same task. There are several open problems involved in scientific workflow construction, sharing and validation. Among them, the paper is concerned with traceability mechanisms.

5.2.2 Related Work

Scientific workflows [106] are being increasingly adopted as a means to specify and coordinate the execution of experiments that involve participants in distinct sites. Such workflows allow the representation and support of complex tasks that use heterogeneous data and software [22]. Bioinformatics is one of the research domains that has widely profited from this facility [52, 82, 99]. However, these systems do not capture provenance information or provide the necessary features to exploit it, hampering the validation and reuse of bioinformatics data.

Workflow and Provenance Representation.

The Workflow Management Coalition (WFMC) [51] has defined a generic model for workflow representation to provide interoperability among different workflow systems. This model has been extended in several ways - e.g. to provide more semantic information or facilitate the sharing of scientific experiments. Also to helps interoperability, many tools invoked by such workflows are now encapsulated into Web services.

In particular, we take advantage of the extension described in [71, 84], which concerns a multi-layered workflow representation to support sharing and reuse at different abstraction levels. Workflows are represented in three layers: abstract workflow (information about the generic structure of a workflow), concrete workflow (instances of the abstract workflow) and executable workflow (a workflow with its input data and parameters).

Several groups have conducted research on provenance representation [75, 84]. A provenance model must represent the “what”, “when” and “where” of information managed in a system. We argue that a provenance model must also be flexible to address distinct levels of detail in workflow representation, and to consider the needs of distinct user groups. Some of our previous work concerns a layered provenance model that considers such flexibility [10, 11].

Our workflow model combines the features of the layered workflow model [71] and the layered provenance model [10, 11]. This combination allows workflow and provenance information to be stored, queried and shared at different abstraction layers. We also added a fourth layer called *run-time workflow* to store information about the executions of an *executable workflow*.

Data Integration and Interoperability.

There are several approaches to deal with data heterogeneity, varying from the conceptual to the physical level [50]. All of these approaches require establishing a mapping among different perspectives of the world. The conceptual level perspective provides to all users a single conceptual model of their applications. This approach has been widely used in bioinformatics frameworks [85, 96, 29]; this kind of solution is useful when the groups involved use the same model, but it does hamper data sharing among groups that use different models. At the other end of the spectrum lies the approach involving structure (and conceptual) mapping [13]. This approach is based on keeping a record of all data transformations that are needed in a system to allow the integration of heterogeneous data from several sources. This record is then updated whenever a new kind of transformation is available.

Our framework is unique in that it uses a structure mapping record combined with ontological description of the concepts to provide a flexible way to integrate heterogeneous data. This combination takes advantage of the relationships present in the ontologies to map concepts into each other, when defined by distinct user groups.

Global conceptual models or structure mapping are common solutions for data integration. Interoperability among systems requires additional approaches, including data exchange standards, use of Web services and interface matching algorithms [50]. The latter are based on trying to match a request for some data or process (caller request) with candidates that best fit this request. Approaches vary according to the compatibility between interfaces – e.g., [88].

Our proposal combines the matching and ranking algorithm from Santanchè and Medeiros [88] with the structure mapping of Bowers and Ludäscher [13], enhanced with workflow composition mechanisms [37, 36]. The goal is to facilitate the integration of data and the interoperability among different sites.

Traceability.

Traceability mechanisms, in the sense used in this paper, are typically found in work that deals with software engineering, electronic commerce or supply chain problems. In software engineering, traceability is used to establish relationships among requirements,

specifications, design and the corresponding code [68]. This allows the verification of, for instance, how requirements evolve to an implemented functionality or to start from a piece of software and trace back the corresponding requirements. In electronic commerce systems, traceability is considered a trust enhancer that is used to trace transactions, payments, and measurements, which provides the assurance of fairness and methods to establish legal proof and redress [97]. In supply chain systems, traceability means providing the identification of all the steps (processes) that were executed to generate a given product, including all the raw matter and intermediate inputs used, the quality of each input, the place where it was produced, etc [83].

Our proposal aims to take advantage of these traceability mechanisms and apply them to bioinformatics, allowing the back- and forward trace of experiments.

5.3 Architecture

We have developed a framework [37, 36] to take advantage of ontologies to support the specification and annotation of bioinformatics workflows. Our framework extends Artificial Intelligence planning techniques with ontologies to support design, reuse and annotation of bioinformatics experiments, specified as scientific workflows. Each activity within such a workflow can be executed either by invocation of a Web service or of another (sub-)workflow. Workflows are stored in databases, and ontology repositories are used to enhance the semantics of automatic bioinformatics workflow construction.

This paper extends this design and management framework with mechanisms to support full traceability of experiments. This approach takes advantage of a workflow provenance model [10, 11] to support efficient storage of provenance information.

Figure 5.1 shows the resulting architecture, portrayed in four layers: Interface, Processing, Data Manager and Repositories. The modules in dark gray correspond to extensions introduced to support traceability. The Interface Layer contains the graphical interface that presents to the user all the functionalities of the framework. The Processing Layer is composed of several processing modules that are responsible for: register and discovery of services, design of workflows (including their automatic composition) and workflow execution. The Data Manager Layer is the middleware between the repositories and the rest of the system. It contains features to support interoperability, data integration and traceability. The Repositories Layer stores information about services, structure types, ontologies, data transformation rules, workflows and provenance data.

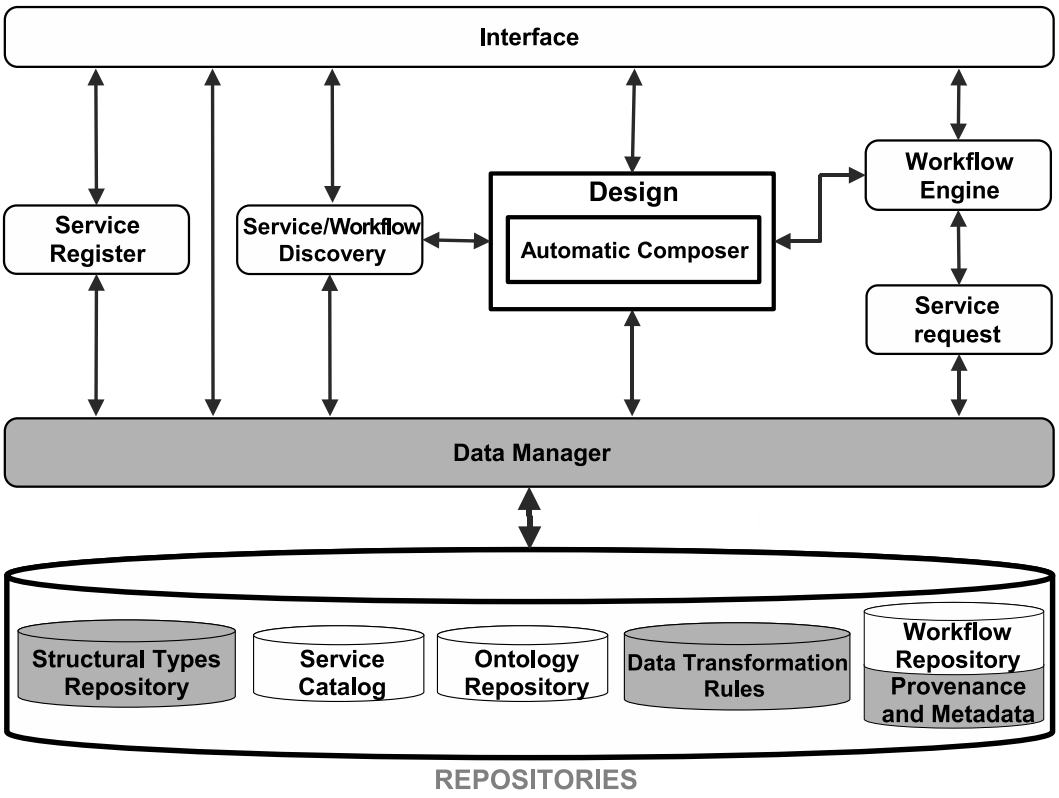


Figura 5.1: System Architecture - dark gray boxes show features to support traceability

5.3.1 Repositories

Five data repositories, stored in a database, serve as the basis for the design, annotation, execution, sharing and traceability of bioinformatics experiments: *Ontology*, *Structural Types*, *Service Catalog*, *Workflow* and *Data Transformation Rules*.

The Ontology Repository contains information about concepts and service types of a given domain. The Structural Types Repository records information about the ways in which the instances of a concept can be stored. The Workflow Repository stores workflows (and parts thereof), in the four abstraction layers described in Section Workflow and Provenance Representation, as well as all data needed to run these workflows. The Workflow Repository also has now been extended to contain provenance information and metadata. The Data Transformation Rules Repository stores information about the valid transformations from a data structure to another.

In more detail, the *Ontology Repository* stores two kinds of ontologies: *Domain Ontology* and *Service Ontology*. The Domain Ontology contains the relationships among the concepts of an application domain (in our case, bioinformatics concepts – e.g., *genome*). The Service Ontology stores the types of services used in this domain, without their instantiation – e.g., *alignment service* or *base–calling service*.

The *Structural Types Repository* specifies the data structures for storing data used in an experiment – e.g., the inputs, outputs, parameters of an experiment. Structural types can be basic (integers, strings, etc) or, recursively, a composition of types.

The *Service Catalog* extends the role of an UDDI (Universal Description, Discovery, and Integration). It stores information about service providers, the Web services available and their interfaces. Furthermore, it also stores non-functional attributes of the service instances, qualifying service interfaces with terms from our Domain Ontology and classifying each operation of the service according to our Service Ontology. An example of a catalog entry is *BLAST*, stating that *BLAST* is of type *alignment service* at URI `xml.nig.ac.jp/wsdl/Blast.wsdl`.

The *Workflow Repository* extends the layered workflow model of [84], adding information about provenance [10, 11]. Such information is stored for each workflow layer; thus, it can refer to an abstract workflow, a specific instance of a workflow, the parameters and input data of a given executable workflow or to the results of a workflow execution (including information about the produced data, steps run, etc). In other words, the Workflow Repository also stores all data needed to execute a workflow.

The *Data Transformation Rules repository* contains three kinds of information: *Default Concept Representation*, *Automatic Transformation* and *Structure Mapping*. The *Default Concept Representation* is a table that links each type in the Structural Types Repository to a concept in the Domain Ontology. That type is considered by the system as the default structural type to represent this ontological concept and it can be customized

by the user. For instance, the entry `<blast_alignment,alignment>` indicates that type `blast_alignment` is the default type to represent an `alignment`. Structure Mapping and Automatic Transformation Rule are tables that contain rules to determine additional transformation across structures in a domain.

Automatic Transformation is a table in which each entry contains a set of rules that determine the automatic execution of workflows to convert *input data* into the desired type. It is stored as <`domain concept, structural type, workflow id, date`>. The `date` indicates when the automatic transformation should be executable (for example, every night, every weekend or every time data is inserted or updated). Following the date restriction, the workflow indicated by the rule will be executed to transform the input data (if it matches the domain concept and the structural type of the rule). For example, the user can insert a rule that says that whenever a *chromatogram* is inserted in the system using the *default chromatogram structural type*, the system must execute a workflow that transforms the *chromatogram* (using a $base - calling$ activity) and produces the *DNA sequence* and the *quality* of the sequence. *Chromatograms* are binary files that contain the raw data of a *DNA sequence*.

Structure Mapping specifies mappings of the possible transformations among structural types in a given domain. Each entry in this table is a tuple of the form <`source type, target type, sub-domain, workflow id, mapping losslessness, mapping classification`> where the mapping from source type to target type is executed by the workflow. The sub-domain delimits the scope in which the transformation can be applied. For example, an *integer* (basic structural type) can always be transformed into a $float$ (another basic structural type), so the sub-domain of this transformation is our full Domain Ontology. On the other hand, a *flat file* can only be transformed into a *xml blast file* if the two structures refer to the concept *blast alignment* from the Domain Ontology. This kind of data transformation mapping is especially useful when considering data and tools that are heterogeneous and from distinct sites.

Each transformation can be classified following two criteria: losslessness and equivalence [88]. A transformation can be *degenerative* (lossy), when there is some loss in data precision (e.g., converting float numbers to integer numbers) or *non-degenerative* (lossless), when there is no loss of information. In terms of equivalence, the results of a transformation can be *equivalent*, *super-structure* (super-type), *sub-structure* (sub-type) or *mixed*. The results are *equivalent* when all information from the source structural type is used (and sufficient) to fill all the information in the target structural type. An example is when the difference between the two types is only the name of a field – such as, if the source structure has a field called *dna_sequence* of type *string* and the target structure has the "same" *string* field but with the name *sequence of DNA*. The result of a transformation is a *super-structure* when the information from the source structure is enough to fill

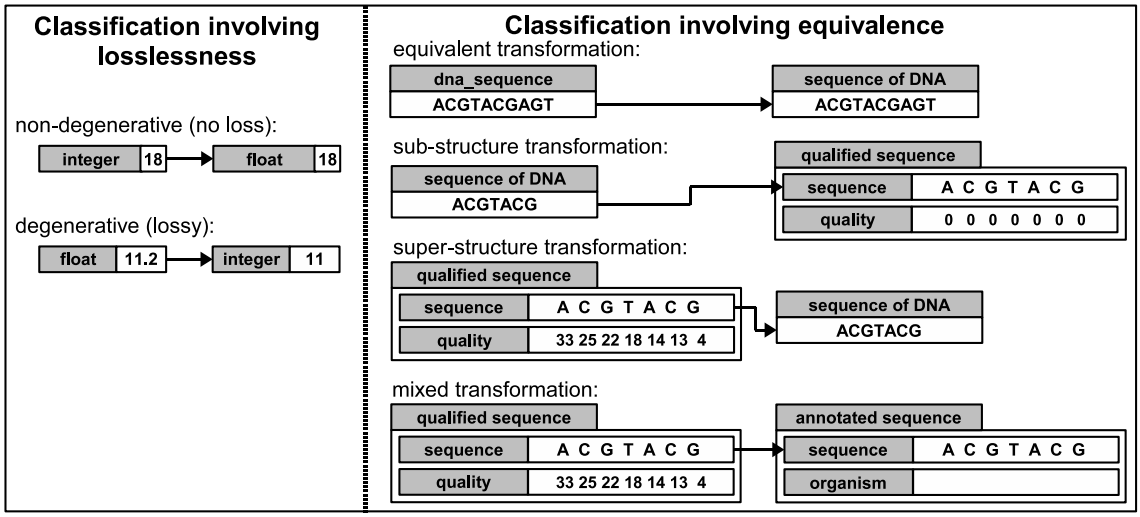


Figura 5.2: Data Transformation Classification

the target structure but some of the source information is not used – e.g., when transforming a structure that has a *DNA sequence* and the *quality* of this sequence into a structure that has only the *DNA sequence*. The result of a transformation is a *sub-structure* when the information from the source structure is totally used in the transformation but it is not enough to fill all the information in the target (e.g., when transforming a structure that has only a *DNA sequence* into a structure that has *DNA sequence* and the *quality* of this sequence), in this case, the missing information can be filled with $default$ or *null* values. The results of a transformation are considered *mixed* when there is information in the source structure that is not used in the target structure and there is information in the target structure that cannot be filled by the source structure. Figure 5.2 illustrates these data transformation examples.

5.3.2 Data Manager

The three main functionalities of the *Data Manager* Layer are:

1. It serves as the middleware (interface) between the other modules and the repositories;

2. It executes the automatic data transformations using the Data Transformation Rules;

3. It supports traceability requests.

The middleware functionality (1) provides data independence - internal storage structures can change without affecting other modules.

The execution of automatic data transformations (2) is useful to preprocess information (accelerating subsequent queries involving this information), to convert data to a desired type, and to facilitate traceability functionalities. For example, suppose a user declares that the input of a workflow in the *Workflow Repository* is a *chromatogram*; there is little information that can be extracted from this, since a chromatogram is a raw binary file. However, if an automatic transformation is executed to transform the chromatogram into a *sequence of nucleotides* with *sequence of quality values*, it is possible to pose queries on sequence alignment, search for patterns, etc.

The third functionality of the Data Manager is to provide traceability mechanisms. To start, it supports basic queries about data and metadata – e.g., "show all sequences from *LBI-UNICAMP*", or "show all *sequence alignments* produced by *blastp* alignment tool". Moreover it allows queries about the process that produced the data, as well as queries on temporal dependencies among activities – e.g. show the experiments that used the *blastn* alignment tool whose input was processed by *phred* base-calling tool.

To allow traceability processing, it also supports an extended set of domain-specific operations, for example, the *sequence alignment operation*. This allows formulation of complex traceability queries – e.g., "select all the *chromatograms* whose *DNA sequence* aligns with a given sequence".

5.3.3 Interactions Between Data Manager and the Other Modules

The Data Manager provides the required functionality for data integration and traceability. It translates requests from the other modules to the data Repositories and vice-versa, providing data independence. This simplified the specification of the other modules of the architecture and ensured it extensibility. Here, we briefly describe the messages exchanged between these modules and the Data Manager. A detailed description of the modules' functionality can be found in [36, 37].

The *Service Register* module lets users register services into the Repository layer, via the Data Manager. The Data Manager provides information about the ontologies that are used to classify the Web service operations. The user must classify each operation offered by a service according to some type (from the Service Ontology) and must assign an ontological concept (from the Domain Ontology) to each operation parameter. Parameters structural types can be deduced from the WSDL specification of the service.

The *Service/Workflow Discovery* module searches for services or workflows that can perform a given task. It starts by requesting from the Data Manager the list of services of a given ontological type from the Service Ontology, and services with a given interface which produce results of a given concept type from the Domain Ontology. Whenever no

suitable service or workflow is found, the Service/Workflow Discovery module notifies the user, who takes the appropriate action – e.g. designing a new workflow. More details can be found in [37, 36].

The *Design* module provides support to workflow design and composition. It asks the Data Manager information about the services and workflows available; the input and output data of a workflow already executed; and sends to Data Manager user updates to a given workflow.

The *Service request* module asks to the Data Manager information about how to invoke a given Web service, such as the provider URL, the messages that will be sent, the structural type of the service answer, etc.

Finally, the user can pose traceability and provenance queries via the interface. These queries are forwarded to the Data Manager, which processes them, and results follow the inverse path.

5.4 Implementation Issues

The first version of the architecture has been implemented as follows. All repositories are stored in a database under the MySQL 5.0 DBMS.

Most other modules are implemented in Java. Design facilities incorporate modules to support workflow composition and edition; one of these modules was specified using SHOP2, an Artificial Intelligence planner. The Data Manager contains the core functionalities to support traceability and provenance tasks. It is also programmed in Java, and acts as an interface between MySQL and the remaining modules.

5.4.1 Domain Specific Query Operators

The Data Manager encapsulates transformation mechanisms that let the other modules pose queries that have domain specific operators – e.g., *base calling*, *alignment*, etc. It translates these queries into a sequence of basic MySQL 5.0 queries, to be executed on the underlying database.

Domain specific operators can be used in two clauses: *SELECT* and *WHERE*. Let *op* be a domain specific operator and y some repository element to which it is meaningful to apply *op*. Thus, *SELECT op(y) FROM ...* first retrieves y from the repositories according to the query predicate in the *WHERE* clause, and then computes *op(y)* – executing *op* on y.

The same kind of procedure is applied in the *WHERE* clause: the predicate to be satisfied is the result of the computation of *op(y)*.

Let us first exemplify the *SELECT* clause. A user wants to see all the *chromatograms* that have been used/referenced in any workflow stored in the *Workflow Repository* – which, we recall, contains workflows at several abstraction levels, data, provenance information, and annotations. However, since chromatograms are stored in binary files, the user wants to see only the *id* of the chromatograms and their *nucleotide sequence*. The *nucleotide sequence* can be produced by the *base_calling* domain specific operator. However, this operator returns objects of type *sequence_and_quality*, which have three attributes – see example of such an object in Figure 5.3. In this case, the user must specify which components of *sequence_and_quality* objects are to be returned, using the standard *dot* notation of query languages. The select statement to execute this query is:

```
SELECT chromatogram.id,
       base_calling(chromatogram).nucleotide_sequence
       FROM WorkflowRepository
```

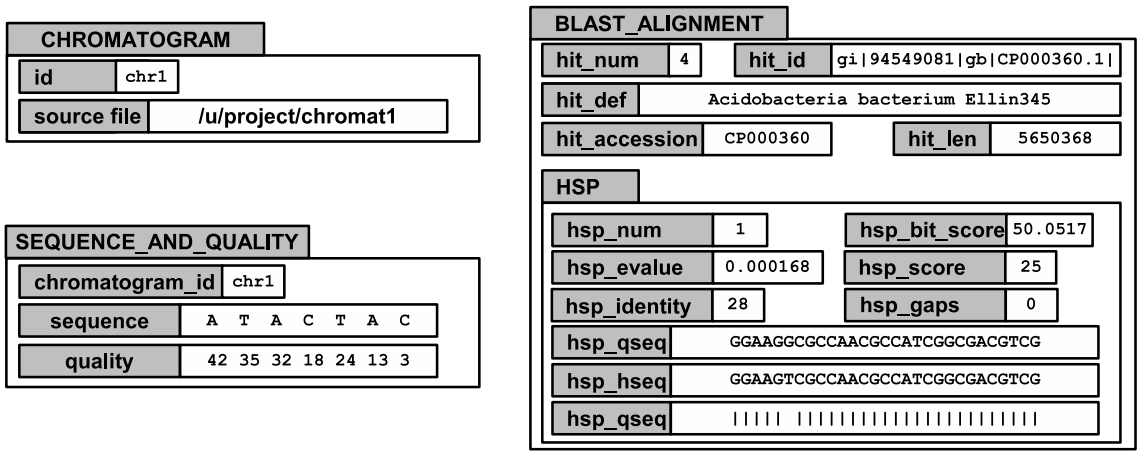


Figura 5.3: Examples of objects of types *chromatogram*, *sequence_and_quality* and *blast-alignment*

Consider now a *WHERE* with special computations. For example, suppose a user wants to retrieve all *reads* that align with a given *read* whose *identity* is greater than or equal to a given value:

```
SELECT read FROM WorkflowRepository WHERE
    alignment(read,'GGAAGGCGCCAACGCCATCGGCGACGTCG').hsp_identity >= 25
```

A *read* is the sequence of letters A, C, G, and T extracted from a chromatogram; *alignment* is a domain specific operator whose result is represented in the form of a *blast_alignment* structure (see Figure 5.3). This operator receives as input two *DNA sequences* (*read* is a sub–type of *DNA sequence*) and produces an object of the type *blast_alignment*. This operator invokes a tool called BLAST (Basic Local Alignment

Search Tool) [6] to execute this operator. Blast_alignment is a structure composed by several fields (based in the XML blast model [67]; we highlight the fields named hit_num (the number of the alignment), the hsp_score (the score of the alignment) and the hsp_identity value (the number of nucleotides that matched in the alignment).

5.4.2 Tool and Data Traceability

The implementation of traceability mechanisms depends on two factors: (i) a storage infrastructure that stores all relevant information about the data and the processes; and (ii) tools to support forward and backward traceability.

Section Repositories presented our storage infrastructure. The second factor depends on the user's needs. This section presents an example of a sequence assembly experiment, for which we discuss six traceability queries to illustrate specific points.

Suppose a user wants to assemble a set of DNA sequences to produce the *assembly_result*. This set is composed of three subsets: `read1` (reads extracted from chromatograms and inserted in the system by the user); `read2` (reads inserted by the user but without corresponding chromatograms) and `contigs` (preassembled reads). After being assembled, not all of the sequences may be present in the *assembly_result* (which always happens in big assemblies). The input sequences subsets can be redistributed in six groups: `read1in`, `read1out`, `read2in`, `read2out`, `contigsIn` and `contigsOut` – where the suffixes mean that the group of sequences is present in the *assembly_result* (`in`) or not (`out`). Note that it is only possible to know if a sequence (or parts thereof) was actually used in the assembly experiment after the execution of the assembly workflow, and then only if the assembly workflow stores (as part of its result) a description of the read composition of the resulting assembled sequence.

Figure 5.4 shows a graphical representation of this experiment. READ1, READ2, and CONTIG correspond to the subsets `read1`, `read2`, and `contigs`, respectively. CHROMATS corresponds to the raw data from the chromatograms (before the *base–calling* operation, invoked to extract chromatogram fields sequence and quality – see Fig. 5.3). The data in the datasets CHROMATS, READ1 and CONTIG was partially inserted by the users and partially produced by the system. To simplify the explanation, let us consider that the CONTIG dataset has only four contigs (called `contig1`, `contig2`, `contig3` and `contig4`). The rectangles in the figure correspond to workflows. There are five workflows: *base–calling wf1*, which converts data from CHROMATS to READ1 *assembly_wf1*, *assembly_wf2*, *assembly_wf3*, which produce, respectively, `contig1`, `contig2` and `contig3`; `contig4` was inserted by the user. *ASSEMBLY WORKFLOW*, the fifth workflow, produces the *assembly_result*. The elements in dark gray are the sequences that are present in the *assembly_result* – that is, they belong to the groups `read1in`, `read2in` and `contigsIn`.

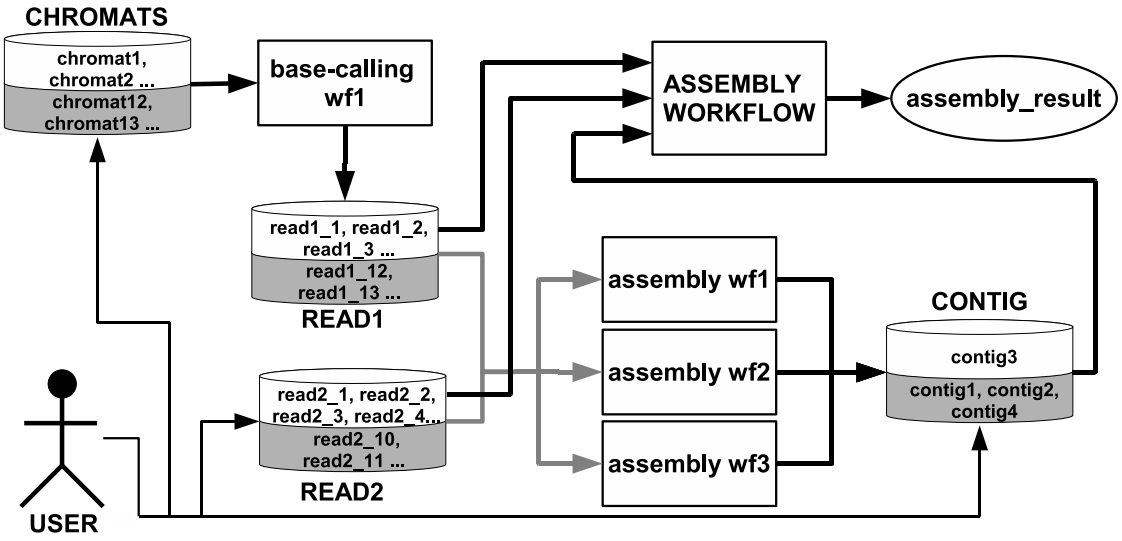


Figura 5.4: Example of an assembly experiment.

When querying the databases, the user can choose between two kinds of answers: nonrecursive (which shows only the immediate result of a query) and recursive (which recursively navigates in an experiment to give a more complete traceability answer). The latter is useful in the case of a workflow with composite activities (i.e., where one activity itself encapsulates a workflow). This supports traceability at distinct abstraction levels. There follows a six representative traceability demands, two of which (1 and 5) involve data used in processes, and the others process execution details.

1. **What is the input data that was used to run the ASSEMBLY WORKFLOW?** The nonrecursive answer is the data from the sets `read1`, `read2`, and `contigs`. The recursive answer also includes the data from CHROMATS that generates the data from the set `read1`, and all the DNA sequences and chromatograms that were used to produce the contigs (from the `contigs` set). This question requires backward tracing and its answer involves all data used in a workflow execution. It is useful when a user intends to re-execute an experiment and needs all the input information to do this.

2. **What are the input sequences that are present in the *assembly_result*?** The answer is equivalent to the previous one, but will contain sequences from `read1in`, `read2in`, and `contigsIn`. This question also requires backward tracing and is useful to distinguish the input data (query 1) from the data actually used.

3. **What processes were used to produce *assembly_result*?** The nonrecursive answer is only *ASSEMBLY WORKFLOW*. The recursive answer also includes *as-*

sembly wf1, *assembly wf2* and *base–calling wf1*. The answer - a set of processes (workflows) - uses backward tracing. Note that `contig3`, the data produced by *assembly wf3*, was not used in the *assembly_result*).

4. **What services/tools were used to produce *assembly_result*?** This requests all the services that compose the workflows from the previous query. It uses backward tracing that, recursively, looks inside each workflow for services used to produce the *assembly_result*.

5. **What data was produced using raw chromatogram number 1 (from the CHROMATS dataset)?** The result is the DNA sequence produced through the use of the *base–calling* workflow and all the assemblies that use this sequence. This needs forward tracing and is useful when a user wants to update a given information and needs to know what data derives from this information.

6. **What services or processes produce an *alignment*?** The answer is the set of all services and workflows whose outputs produce an *alignment* or any sub–concept (in the Domain Ontology) of *alignment*. This requires a combination of interface matching and semantic relationships, being useful when a user wants to know what tools can produce a desired concept (or data type).

We point out that we provide traceability functionality that is normally unavailable in standard mechanisms. This is achieved thanks to the fact that we store extended semantic knowledge about each tool and process used in an experiment.

For instance, query 2 wants to know which of the input sequences effectively contributed to construct the result. This cannot be deducted in standard representations, where the only information available is some kind of tuple <`input data, process, output`>. Our repository management mechanisms instead, annotate processes and tools with a sophisticated type facility. Here, for instance, our system knows that this execution of *Assembly Workflow* invoked a specific tool whose results include logs of its intermediate steps. The processing of query 2 is thus transformed into a set of queries that first identify the type of the tool used in the assembly and, afterward, process all of its logs, themselves annotated with structural types and ontology terms.

5.5 Conclusions

This paper presented a framework to support bioinformatics experiments enhanced with traceability mechanisms. This framework helps scientists to record detailed information about all processes involved in an experiment. Our solution takes advantage of structure

mapping to facilitate data integration, and of interface matching to provide interoperability. It also extends database query facilities to support domain-specific concerns. The framework has been validated by a prototypical implementation with real data.

Several bioinformatics laboratories use a scientific workflow infrastructure to design and execute their experiments - e.g., [52, 82, 99]. Our work extends these approaches through the use of the Data Manager layer that, when combined with data repositories, provides interoperability, data integration and traceability mechanisms.

As future work we intend to specify and implement a ranking algorithm to sort the results of a query, based on the suitability of each result. We also intend to develop mechanisms for sharing transformation rules. Finally, we intend to develop graphical tools to facilitate data presentation and include user feedback loops to improve retrieval facility.

Capítulo 6

Conclusões

6.1 Contribuições

Esta tese apresentou uma infra-estrutura para gerenciamento de experimentos em bioinformática. A pesquisa realizada combinou aspectos de bancos de dados, planejamento em Inteligência Artificial, gerenciamento de ontologias, padrões da Web Semântica e workflows científicos. Como resultado, apresentou soluções para os desafios de facilitar a especificação, reuso, documentação, validação e compartilhamento de experimentos científicos de bioinformática.

A infra-estrutura especificada e prototipada fornece mecanismos de integração de dados, composição de serviços e rastreabilidade para facilitar o gerenciamento de experimentos científicos. As principais contribuições desta tese são, desta forma:

- Proposta de uma solução para o problema da composição de serviços, combinando resultados da IA e de Bancos de Dados, de forma a ajudar o projeto de workflows científicos e, ao mesmo tempo, documentar alternativas de projeto;
- Definição de um modelo de dados que combine o armazenamento de workflows em camadas com o armazenamento de proveniência de dados e ferramentas;
- Uso de repositórios de ontologias para enriquecer a semântica na construção automática de workflows e facilitar o rastreamento da proveniência de dados e procedimentos;
- Especificação de mecanismos de integração de dados, interoperabilidade de ferramentas e rastreabilidade de experimentos científicos;
- Validação de parte da solução proposta pela implementação de três protótipos em e-Science, um dos quais para a montagem e anotação de genomas.

A solução apresentada foi projetada para problemas de montagem e anotação de genomas. Porém, ela pode ser utilizada em outros domínios bastando para isso: (i) a construção de ontologias de domínio e de serviços apropriadas; e (ii) o desenvolvimento de operadores específicos ao domínio adotado. Em particular, como apresentado na Seção 6.3, aplicamos um dos protótipos à bio-medicina e outro a oceanografia.

6.2 Considerações adicionais

Diversas dificuldades foram encontradas durante o desenvolvimento desta tese, principalmente na implementação do protótipo. Destacamos nesta seção algumas preocupações que julgamos relevantes para especificar ou implementar sistemas para a e-Science.

Ontologias e estruturas de dados. Iniciamos a especificação de nossa ontologia de domínio sem considerar os detalhes das estruturas de dados que seriam anotadas segundo esta ontologia. Este é o caminho mais comum na especificação de ontologias. Porém, a ontologia precisou ser atualizada diversas vezes, pois parte dos dados utilizados nos experimentos de implementação não tinha correspondência com nenhum termo da ontologia. Isto ocorreu principalmente porque, quando se analisa um objeto complexo (como um resultado de um alinhamento produzido pela ferramenta BLAST) há diversos campos cujo conteúdo não está diretamente ligado aos principais conceitos do domínio. Neste tipo de situação há três alternativas principais: (i) ignorar alguns campos e só trabalhar com os conceitos (e dados) considerados principais para o domínio em que se está trabalhando; (ii) estender a ontologia de forma a permitir a anotação de todos os dados (note que praticamente sempre que um novo serviço for inserido no sistema será necessário estender a ontologia); ou (iii) usar uma solução intermediária, que combina os outros dois enfoques. A tese utilizou a terceira alternativa, após algumas frustrações tentando utilizar a segunda.

Uso de serviços Web. A descrição dos serviços Web presente nos arquivos WSDL deve ser aproveitada na hora de cadastrar os serviços dentro do sistema de workflows. De um modo geral, os tipos de dados de todos os parâmetros das operações providas por um serviço podem ser capturados automaticamente a partir do WSDL. Porém, destacamos três cuidados que o desenvolvedor do sistema deve ter: (i) o WSDL nem sempre é totalmente compatível com o serviço Web (às vezes a interface dos serviços é atualizada sem que a especificação WSDL o seja, ou o arquivo em WSDL foi escrito manualmente e contém erros); (ii) o serviço e o arquivo WSDL correspondente podem ter sido atualizados desde a última vez que o sistema de workflows o consultou; e (iii) resultados complexos (por exemplo, contendo diversos campos) muitas vezes são retornados pelos serviços Web na forma de uma *string* com algum tipo de separador de campos; isto pode dificultar a integração desse resultado com o restante do sistema. Este último cuidado é abordado a

seguir, na discussão sobre integração de dados.

Integração de dados. Um dos principais problemas práticos que enfrentamos foi a heterogeneidade com que dados referentes a um mesmo conceito da ontologia são providos por cada serviço ou ferramenta. Nosso primeiro protótipo não tratava aspectos de integração de dados, considerando apenas a correspondência ontológica entre as entradas e saídas dos serviços. Desta forma, os workflows gerados precisavam ser ajustados pelo usuário para se tornarem efetivamente executáveis. Isto feria um dos objetivos desta tese que é facilitar a especificação dos experimentos. Para evitar este tipo de problema, especificamos mecanismos para facilitar a integração de dados combinando o mapeamento de estruturas de dados e o mapeamento conceitual.

Modelo genérico para armazenamento de workflows, dados de proveniência e metadados. Nesta tese definimos um modelo genérico, em quatro camadas para o armazenamento dos dados relacionados a experimentos científicos. O modelo é genérico, porém é importante notar que as funcionalidades relacionadas a proveniência e rastreabilidade dependem do tipo de dado que está sendo armazendo e, conseqüentemente, da comunicação entre o motor de execução de workflows e o restante do sistema.

Funcionalidades do planejador. A ligação entre o planejador utilizado (SHOP2) e as ontologias é uma opção lógica devido à natureza hierárquica dessas duas entidades. A utilização de ontologias em planejadores não hierárquicos é uma atividade mais complexa e não foi abordada nesta tese. Além disso, relacionamentos complexos de precedência entre serviços (tarefas dos planos) também não foram estudados. A implementação realizada não interferiu no funcionamento do planejador, acrescentando no entanto algumas regras adicionais para considerar objetos complexos (objetos que podem ser formados por outros objetos e que podem herdar características de outros objetos).

Protótipos. Durante o desenvolvimento desta tese, implementamos três protótipos para a validação das soluções propostas. A Seção 6.3 descreve esses protótipos. Apenas o terceiro protótipo está sendo utilizado por usuários em ambientes reais (no caso, oceanógrafos). Existem alguns sistemas gerenciadores de workflows que devem ser considerados para o projeto e execução de workflows (exemplo, [5, 101]). Esses sistemas não possuem todas as funcionalidades apresentadas nesta tese (como por exemplo, a composição automática de tarefas), porém são extensíveis e possuem versões estáveis disponíveis na Internet. A utilização efetiva da arquitetura proposta exige desenvolvimento de módulos para sua integração a alguns desses sistemas gerenciadores de workflows.

6.3 Aspectos de implementação

Implementamos três protótipos no decorrer desta tese para testar diferentes aspectos de nossa solução. Nesta seção descrevemos as características de cada um deles e a tecnologia

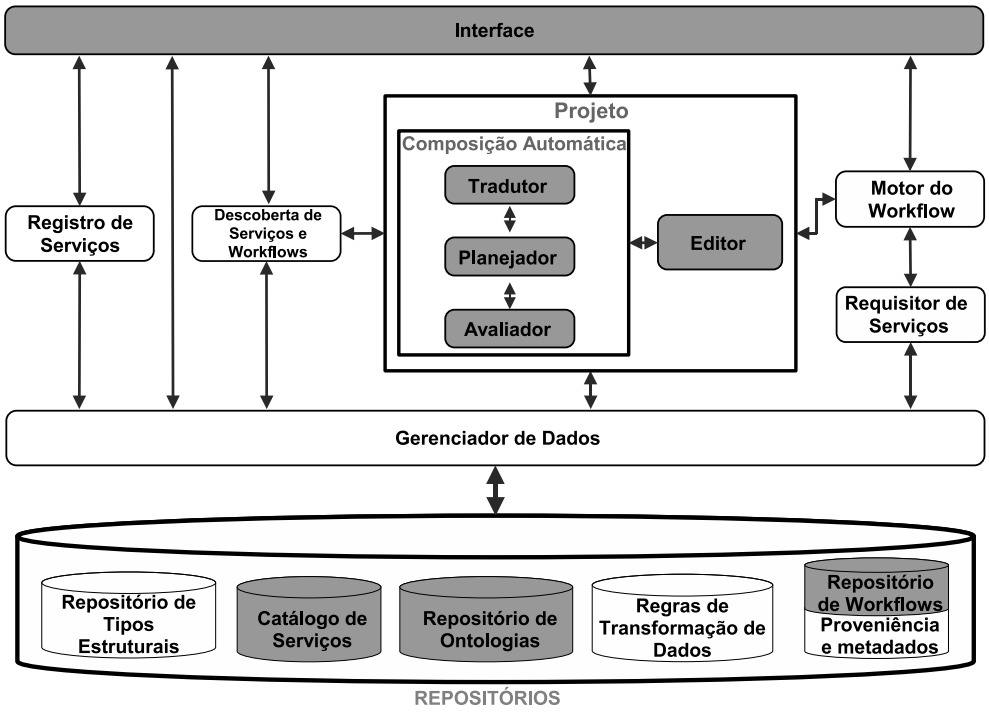


Figura 6.1: Protótipo 1 - Extensão do WOODSS para planejamento

adotada.

6.3.1 Extensão do WOODSS - composição automática

Estendemos uma versão do sistema gerenciador de workflows WOODSS [71] para tirar vantagens do uso de ontologias e do planejamento em IA como mecanismo de composição automática. Os módulos em cinza na Figura 6.1 correspondem aos módulos implementados neste protótipo.

A interface gráfica do protótipo foi desenvolvida em Java (utilizando-se a biblioteca *swing*). O planejador utilizado foi o SHOP2, sendo que a definição de domínio do planejador é feita em Lisp. Construímos *scripts* em Perl e *bash* para ligar o planejador aos demais módulos do sistema. As ontologias de serviços e de domínio foram especificadas em OWL [23], usando a ferramenta Protégé [59] e as ligações entre as instâncias dos serviços e as ontologias foram descritas em OWL-S [69]. O Capítulo 4 mostra exemplos de tela dos workflows gerados a partir da interação com o módulo de composição automática.

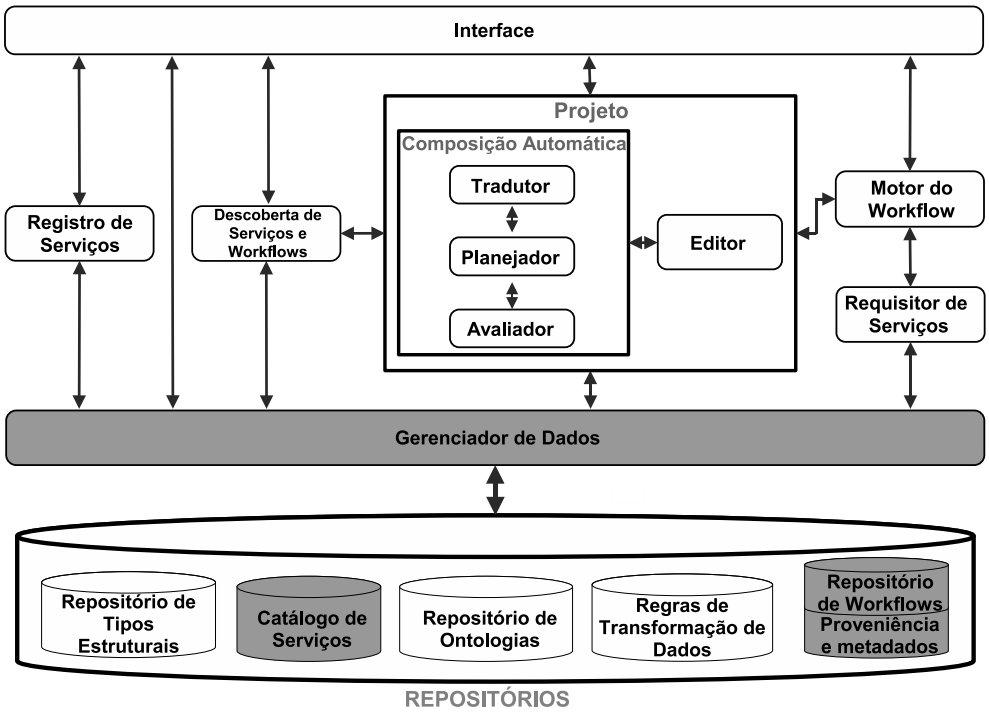


Figura 6.2: Protótipo 2 - Banco de dados de workflows e rastreabilidade

6.3.2 Banco de dados e rastreabilidade

O segundo protótipo corresponde ao banco de dados e sistema de consultas REDUX [10] que foi especificado e implementado durante o nosso primeiro estágio internacional na Microsoft Research. Os módulos em destaque na Figura 6.2 foram implementados neste protótipo, cujo objetivo principal foi testar experimentos em rastreabilidade.

Além da especificação e implementação do banco de dados, que foi testado com dados biomédicos reais [10], também implementamos mecanismos de consulta a proveniência e rastreabilidade. Utilizamos o sistema gerenciador de banco de dados SQLServer 2005 e a linguagem de programação C# para implementar a ferramenta de processamento de consultas.

6.3.3 Projeto e execução remotos de workflows

O terceiro protótipo foi desenvolvido para testar o projeto e a execução de workflows utilizando um navegador de Internet. Ele foi implementado durante o nosso segundo estágio internacional e está sendo testado por oceanógrafos da Universidade de Washington. Os módulos implementados neste protótipo estão destacados em cinza na Figura 6.3.

Este protótipo utiliza o Windows Workflow Foundation [8] como motor de execução. O

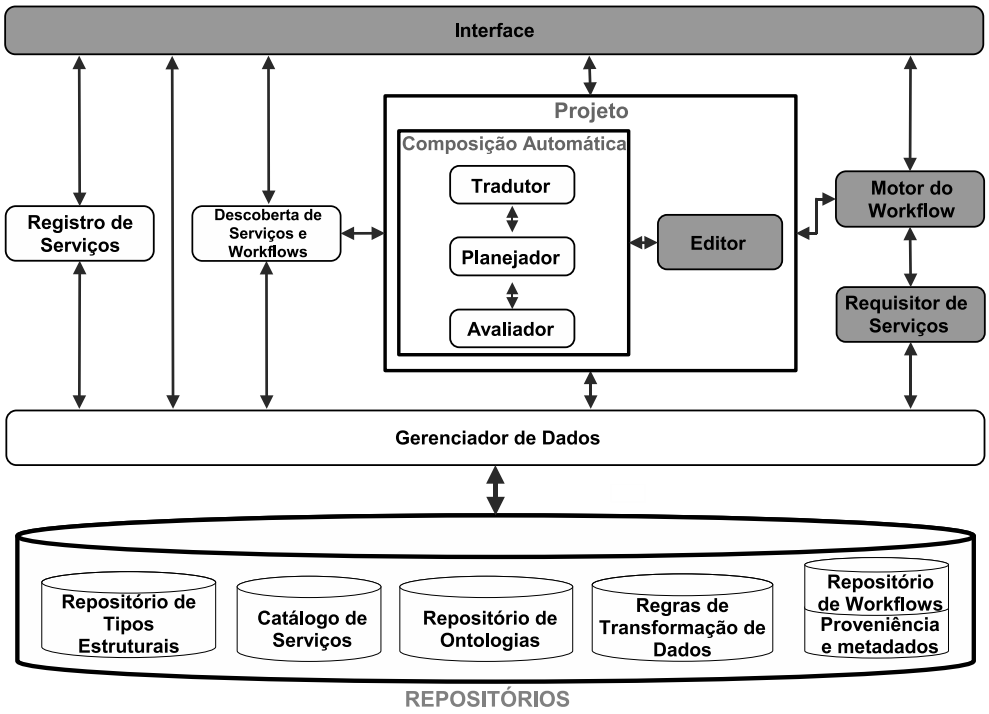


Figura 6.3: Protótipo 3 - Projeto e execução remotos de workflows

editor de workflows foi implementado em C# utilizando-se a tecnologia ASPX. O servidor Web utilizado foi o Internet Information Service 7.0 [55].

6.4 Possíveis Extensões

Há diversos trabalhos futuros previstos para esta tese. Dentre eles, há trabalhos teóricos de pesquisa, extensão da infra-estrutura adicionando novas funcionalidades e especificação de ontologias mais abrangentes ou que envolvam um domínio de aplicação diferente.

Dentre as extensões teóricas de pesquisa, destacamos:

- Avaliação de outros tipos de planejadores além dos hierárquicos (por exemplo, planejadores baseados em Checagem Simbólica de Modelos [16]);
- Definição de mecanismos para a avaliação de planos nos níveis abstrato e concreto;
- Estudo de mecanismos para otimização na geração e execução de planos, tais como re-planejamento e reparo de planos;
- Especificação de mecanismos para a execução eficiente dos workflows (paralelismos, *caches*, etc). Note que esta extensão depende do tipo de motor de workflow utilizado;

- Pesquisa sobre mecanismos de indexação de serviços e workflows.
- Extensão da infra-estrutura proposta para tratar alinhamento e composição de ontologias;

Dentre as extensões aplicadas, destacamos:

- Desenvolvimento de ferramentas gráficas para facilitar a apresentação de dados complexos;
- Desenvolvimento de técnicas de reconhecimento de planos para auxiliar a especificação de workflows e para melhor organizar e compactar os workflows armazenados no banco de dados;
- Extensão do escopo das ontologias para que descrevam contextos mais amplos em bioinformática, tais como genômica comparativa e vias metabólicas;
- Aplicação da infra-estrutura em outros domínios, por exemplo, em geoprocessamento;
- Validação dos propótipos pela execução de testes com usuários.

Referências Bibliográficas

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

[2] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[3] V. Agarwal, G. Chafle, K.A. Kumar, S. Mittal, and B. Srivastava. Synthy: A System for End to End Composition of Web Services. *Journal of Web Semantics*, 3(4):311–339, 2005.

[4] G. Alonso, F. Casati, H. Kuno, and V. Machiaraju, editors. *Web Services: Concepts, Architectures and Applications*. Springer, Heidelberg, German, 2004.

[5] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th Intl. Conference on Scientific and Statistical Database Management(SSDBM)*, Santorini Island, Greece, June 2004.

[6] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[7] M. Altunay, D. Colonnese, and C. Warade. A framework for deploying bioinformatics applications as high-throughput Web services on the NC BioGrid. http://www.ibm.com/developerworks/webservices/library/ws-bioinfo.html (as of 2007-01-19), 2004.

[8] P. Andrew et al. *Presenting Windows Workflow Foundation*. SAMS Publishing, 2006.

[9] R.S. Barga and L.A. Digiampietri. Automatic generation of workflow provenance. In *Provenance and Annotation of Data International Provenance and Annotation*

Workshop (IPAW), volume 4145 of *Lecture Notes in Computer Science*. Springer, 2006.

[10] R.S. Barga and L.A. Digiampietri. Redux - First provenance challenge. http://twiki.ipaw.info/bin/view/Challenge/REDUX (as of 2007-04-04), 2006.

[11] R.S. Barga and L.A. Digiampietri. Automatic capture and efficient storage of escience experiment provenance. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[12] W. Bausch, C. Pautasso, R. Schaeppi, and G. Alonso. Bioopera: Cluster-aware computing. In *Proceeding of the the 4th IEEE International Conference on Cluster Computing (Cluster)*, pages 90–94, 2001.

[13] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In Erhard Rahm, editor, *Proceedings of Data Integration in the Life Sciences*, volume 2994 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[14] S. Bowers, T. McPhillips, and B. Ludaescher. A provenance model for collection-oriented scientific workflows. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[15] K.R. Braghetto. *Padrões de Fluxos de Processos em Banco de Dados Relacionais*. PhD thesis, IME-USP, São Paulo, SP, Brazil, 2006.

[16] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: $10\hat{2}0$ States and Beyond. *Information and Computation*, 98(2):142–170, 1992.

[17] D. Buttler, M. Coleman, T. Critchlow, R. Fileto, W. Han, C. Pu, D. Rocco, and L. Xiong. Querying multiple bioinformatics information sources: can semantic web research help? *ACM SIGMOD Record*, 31(4):56–64, 2002.

[18] Cambridge Dictionaries Online. http://dictionary.cambridge.org (as of 2006-12-18).

[19] M. Cannataro, C. Comito, A. Guzzo, and P. Veltri. Integrating ontology and workflow in PROTEUS, a grid-based problem solving environment for bioinformatics. In *Proceeding of the International Conference on Coding and Computing*, pages 90–94, 2004.

[20] M.F. Carazzolle, E.F. Formighieri, L.A. Digiampietri, M.R.R. Araujo, G.G.L. Costa, and G.A.G. Pereira. Gene projects: a web application for ongoing annotation in est and shotgun genome projects. *Genetics and Molecular Biology*, 2007. Accepted for publication.

[21] J. Cardoso and A.P. Sheth. Semantic E-workflow composition. *Journal of Intelligent Information Systems*, 21(3):191–225, 2003.

[22] M.C. Cavalcanti, R. Targino, F. Baiao, S.C. Rössle, P.M. Bisch, P.F. Pires, M.L.M. Campos, and M. Mattoso. Managing structural genomic workflows using Web services. *Data & Knowledge Engineering*, 53(1):45–74, 2005.

[23] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html (as of 2007-01-18), 1999.

[24] The Workflow Management Coalition. Workflow management coalition terminology & glossary (issue 3.0) 1999. http://www.wfmc.org/standards/docs (as of 2005-09-20).

[25] The Workflow Management Coalition. The Workflow Reference Model. Technical Report TC-1003, The Workflow Management Coalition, 1995.

[26] Microsoft Corporation. Microsoft research databases group. http://research.microsoft.com/db/ (as of 2007-02-05).

[27] G.G.L. Costa, L.A. Digiampietri, E.H. Ostroski, and J.C. Setubal. Evaluation of graph based protein clustering methods. In *Proceedings of the Fifth Brazilian Symposium on Mathematical and Computational Biology (BIOMAT2005)*, 2005.

[28] L.A.G. da Costa, P.F. Pires, and M. Mattoso. Automatic Composition of Web Services with Contingency Plans. In *Proceedings of the IEEE ICWS 2004*, pages 454–461, 2004.

[29] M. Diehn, G. Sherlock, G. Binkley, H. Jin, J.C. Matese, T. Hernandez-Boussard, C.A. Rees, J.M. Cherry, D. Botstein, P.O. Brown, and A.A. Alizadeh. SOURCE: a unified genomic resource of functional annotations, ontologies, and gene expression data. *Nucleic Acids Research*, 31(1):219–223, 2003.

[30] L.A. Digiampietri, C. B. Medeiros, and J. C. Setubal. A data model for comparative genomics. *Revista Tecnologia da Informação*, 3(2):35–40, 2003.

[31] L.A. Digiampietri, C.B. Medeiros, and J.C. Setubal. A framework based in Web services orchestration bioinformatics workflow management. In *Proceedings of the Third Brazilian Workshop on Bioinformatics*, pages 17–24, 2004.

[32] L.A. Digiampietri, C.B. Medeiros, and J.C. Setubal. A framework based in Web services orchestration bioinformatics workflow management. *Genetics and Molecular Research*, 4(3):535–542, 2005.

[33] L.A. Digiampietri, C.B. Medeiros, and J.C. Setubal. A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows. In *Proceedings of the Second Ongoing PhD Thesis of IC - UNICAMP*, number IC-06-24, pages 20–22, December 2006.

[34] L.A. Digiampietri, C.B. Medeiros, J.C. Setubal, and R.S. Barga. Traceability Mechanisms for Bioinformatics Scientific Workflows. In *Proceedings of the AAAI2007's Workshop on Semantic e-Science (SeS07)*, pages 26–33, Vancouver, Canada, 2007.

[35] L.A. Digiampietri, J.J. Pérez-Alcazar, and C.B. Medeiros. AI Planning in Web Services Composition: a review of current approaches and a new solution. In *VI ENIA - Proceedings of the XXVII Brazilian Computer Society Conference (CSBC2007)*, July 2007.

[36] L.A. Digiampietri, J.J. Pérez-Alcázar, and C.B. Medeiros. An ontology-based framework for bioinformatics workflows. *International Journal of Bioinformatics Research and Applications*, 3(3):268–285, 2007.

[37] L.A. Digiampietri, J.J. Pérez-Alcazar, C.B. Medeiros, and J.C. Setubal. A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows. Technical Report IC-06-04, Institute of Computing, University of Campinas, February 2006.

[38] L.A. Digiampietri, J. C. Setubal, and C. B. Medeiros. Bioinformatics scientific workflows: combining databases, AI and Web services. In *Proceedings of V Workshop de Teses e Dissertações em Bancos de Dados (WTDBD)*, pages 2–9, 2006.

[39] L.A. Digiampietri et al. Fact and Task Oriented System for genome assembly and annotation. In *Proceedings of the Brazilian Symposion on Bioinformatics*, volume 2594 of *Lecture Notes in Bioinformatics*, pages 238–241. Springer, 2005.

[40] J.D. Eckart and B.W.S. Sobral. A life scientist's gateway to distributed data management and computing: The pathport/toolbus framework. *OMICS*, 7(1):79–88, 2003.

[41] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Research*, 8(3):186–194, March 1998.

[42] R. Fileto, C.B. Medeiros, L. Liu, C. Pu, and E. Assad. Using Domain Ontologies to help Track Data Provenance. In *Proceedings of Brazilian Database Conference, SBBD*, pages 84–98, 2003.

[43] First Provenance Challenge. http://twiki.ipaw.info/bin/view/Challenge/ (as of 2006-11-13).

[44] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 1999.

[45] K. Fujii and T. Suda. Dynamic Service Composition Using Semantic Information. In *Proceedings of ICSOC 2004*, pages 39–48, 2004.

[46] H.T. Gao, J.H. Hayes, and H. Cai. Integrating Biological Research through Web Services. *IEEE Computer*, 38(3):26–31, 2005.

[47] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning, Theory and Practice*. Elsevier, 2004.

[48] D. Gordon, C. Abajian, and P. Green. Consed: A graphical tool for sequence finishing. *Genome Research*, 8(3):195–202, 1998.

[49] D. Hall, J.A. Miller, J. Arnold, K.J. Kochut, A.P. Sheth, and M. Weise. Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequencing Initiative. *Bioinformatics Journal*, 1999.

[50] T. Hernandez and S. Kambhampati. Integration of biological sources: Current systems and challenges ahead. *SIGMOD Record*, 33(3):51–60, 2004.

[51] D. Hollingsworth. The Workflow Reference Model. Technical Report TC-1003, Workflow Management Coalition, 1995.

[52] S. Hoon, K.K. Ratnapu, J. Chia, B. Kumarasamy, X. Juguang, M. Clamp, A. Stabenau, S. Potter, L. Clarke, and E. Stupka. Biopipe: A flexible framework for protocol-based bioinformatics analysis. *Genome Research*, 13(8):1904–1915, 2004.

[53] IBM. Web Service for Bioinformatic Analysis Workflow. http://www.alphaworks.ibm.com/aw.nsf/reqs/wsbaw (as of 2007-01-18), 2003.

[54] *Workshop on Planning and Scheduling for Web and Grid Services*, Whistler, British Columbia, Canada, June 2004.

[55] Internet information service. http://www.iis.net/ (as of 2007-04-04), 2007.

[56] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *KER*, 18(1):1–31, 2003.

[57] J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[58] K.H. Kim. Workflow dependency analysis and its implications on distributed workflow systems. In *Proceedings of the 17th International Conference on Advanced Information Networking and Applications*, pages 677–682, 2003.

[59] H. Knublauch, R.W. Fergerson, N.F. Noy, and M.A. Musen. The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, pages 229–243, 2004.

[60] A. Krenek, J. Sitera, L. Matyska, F. Dvorak, M. Mulac, M. Ruda, and Z. Salvet. glite job provenance – a job-centric view. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[61] Laboratory for Bioinformatics, Institute of Computing, University of Campinas. http://www.lbi.ic.unicamp.br (as of 2005-09-15).

[62] Laboratory for Genomics and Expression, Institute of Biology, University of Campinas. http://www.lbi.ic.unicamp.br (as of 2006-11-23).

[63] Laboratory of Information Systems, Institute of Computing, University of Campinas. http://www.lis.ic.unicamp.br (as of 2007-02-08).

[64] H.J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R.B. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–84, 1997.

[65] D. Long and M. Fox. The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

[66] P. Lord, S. Bechhofer, M.D. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *Proceedings of the the 3rd International Semantic Web Conference*, pages 350–364, november 2004.

[67] T. Madden. The blast sequence analysis tool. http://www.ncbi.nlm.nih.gov/books/bookres.fcgi/handbook/ch16d1.pdf (as of 2007-01-19), 2002.

[68] J.I. Maletic, M.L. Collard, and B. Simoes. An XML based approach to support the evolution of model-to-model traceability links. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 67–72, New York, NY, USA, 2005.

[69] D. Martin et al. OWL-S: Semantic Markup for Web Services, 2004. http://www.daml.org/services/owl-s/1.1B/owl-s.pdf (as of 2005-09-08).

[70] S.A. McIlraith and T.C. Son. Adapting Golog for Composition of Semantic Web Services. In *KR2002*, pages 482–493, 2002.

[71] C.B. Medeiros, J. Perez-Alcazar, L. Digiampietri, G. Pastorello, A. Santanche, R. Torres, E. Madeira, and E. Bacarin. WOODSS and the Web: Annotating and Reusing Scientific Workflows. *ACM SIGMOD Record*, 34(3):18–23, 2005.

[72] B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing web services on the semantic web. *VLDB Journal*, 12(4):333–351, 2003.

[73] J. Meidanis, G. Vossen, and M. Weske. Using workflow management in DNA sequencing. In *CoopIS*, pages 114–123, 1996.

[74] S. Miles, P. Groth, S. Munroe, S. Jiang, T. Assandri, and L. Moreau. Extracting Causal Graphs from an Open Provenance Data Model. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[75] L. Moreau and I.T. Foster, editors. *Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*, volume 4145 of *Lecture Notes in Computer Science*. Springer, 2006.

[76] L. Moreau and B. Ludäscher. The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[77] L.M. Moreira, R.F. de Souza, L.A. Digiampietri, A.C. R. da Silva, and J.C. Setubal. Comparative analyses of xanthomonas and xylella complete genomes. *OMICS: A Journal of Integrative Biology*, 9(1):43–76, 2005.

[78] H. Munoz-Avila, D.W. Aha, D. Nau, R. Weber, L. Breslow, and F. Yaman. SiN: Integrating case-based reasoning with task decomposition. In *Proceedings of IJCAI 2001*, pages 999–1004. Morgan Kaufmann, 2001.

[79] A.L.T.O. Nascimento, S. Verjovski-Almeida, M.A. Van Sluys, C.B. Monteiro-Vitorello, L.E.A. Camargo, L.A. Digiampietri, R.A. Harstkeerl, P.L. Ho, M.V. Marques, M.C. Oliveira, J.C. Setubal, D.A. Haake, and E.A.L. Martins. Genome features of leptospira interrogans serovar copenhageni. *Brazilian Journal of Medical and Biological Research*, 37(4):459–477, 2004.

[80] D. Nau, T.C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[81] D. Nau, S. Gupta, and W. Regli. Artificial Intelligence Planning versus manufacturing-operation planning: a case study. In *Proceedings of IJCAI 1995*, pages 1670–1676. Morgan Kaufmann, 1995.

[82] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[83] L.U. Opara. Traceability in agriculture and food supply chain: a review of basic concepts, technological implications, and future prospects. *Food, Agriculture & Environment*, 1(1):123–125, 2003.

[84] G.Z. Pastorello Jr., C.B. Medeiros, S.M. Resende, and H.A. Rocha. Interoperability for GIS Document Management in Environmental Planning. In *Journal of Data Semantics*, number 3534 in Lecture Notes in Computer Science, pages 100–124. Springer, 2005.

[85] J.D. Peterson, L.A. Umayam, T. Dickinson, E.K. Hickey, and O. White. The comprehensive microbial resource. *Nucleic Acids Research*, 29(1):123–125, 2001.

[86] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. In *SWSWPC 2004*, volume 3387, pages 43–54, 2004.

[87] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, USA, 2003.

[88] A. Santanchè and C.B. Medeiros. Self describing components: Searching for digital artifacts on the web. In *Proceedings of Brazilian Symposium on Databases (SBBD)*, pages 10–24, 2005.

[89] C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 2007. To appear.

[90] L. Seffino, C.B. Medeiros, J. Rocha, and B. Yi. WOODSS - A Spatial Decision Support System based on Workflows. *Decision Support Systems*, 27(1-2):105–123, 1999.

[91] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, 1997.

[92] A.J.G. Simpson et al. The genome sequence of the plant pathogen Xylella fastidiosa. *Nature*, 406(1):151–157, 2000.

[93] E. Sirin, B. Parsia, D. Wu, J. A. Hendler, and D. S. Nau. HTN planning for Web Service composition using SHOP2. *Journal of Web Semantics*, 1(4):377–396, 2004.

[94] B. Smith, J. Williams, and S. Schulze-Kremer. The ontology of the gene ontology. In *Proceeding of AMIA 2003*, pages 609–613, 2003.

[95] B. Srivastava and J. Koehler. Web Service Composition - Current Solutions and Open Problems. In *ICAPS 2003*, pages 28–35, June 2003.

[96] L.D. Stein, C. Mungall, S.Q. Shu, M. Caudy, M. Mangone, A. Day, E. Nickerson, J.E. Stajich, T.W. Harris, A. Arva, and S. Lewis. The generic genome browser: A building block for a model organism system database. *Genome Research*, 12(10):1599–1610, 2002.

[97] D.D. Steinauer, S.A. Wakid, and S. Rasberry. Trust and traceability in electronic commerce. In *Proceedigns of StandardView*, volume 5, pages 118–124, 1997.

[98] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2):184–186, 2000.

[99] R.D. Stevens, H.J. Tipney, C.J. Wroe, T.M. Oinn, M. Senger, P.W. Lord, C. A. Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren syndrome using myGrid. *Bioinformatics*, 20(1):i303–310, 2004.

[100] K.P. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *J. Web Sem*, 1(1):27–46, 2003.

[101] The myGrid Consortium. myGrid: Middleware for in silico experiments in biology. http://www.mygrid.org.uk/ (as of 2005-07-11).

[102] P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. *Lecture Notes in Computer Science*, 3298:380–394, 2004.

[103] M.A. Vouk. Integration of heterogeneous scientific data using workflows - a case study in bioinformatics. In *Proceedings of the 25th International Conference on Interfaces*, volume 16–19, pages 25–28, 2003.

[104] W3C. Web Service Choreography Interface (WSCI) 1.0. http://www.w3.org/TR/wsci (as of 2007-01-18), 2002.

[105] W3C. Business Process Execution Language for Web Services Version 1.1. http://www.w3.org/International/ws/ws-i18n-scenarios-edit/ws-i18n-requirements-edit.html (as of 2007-01-18), 2003.

[106] J. Wainer, M. Weske, G. Vossen, and C.B. Medeiros. Scientific Workflow Systems. In *Proceedings of the NSF Workshop on Workflow and Process Automation Information Systems*, 1996.

[107] I. Yoo and X. Hu. Biomedical ontology mesh improves document clustering qualify on medline articles: A comparison study. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pages 577–582, Los Alamitos, CA, USA, 2006.

[108] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 34(3):44–49, 2005.

[109] J. Zhao, C. Goble, R. Stevens, and D. Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 2007. To appear.