3
Projeto de Proveniência

Neste capítulo será descrito o projeto de proveniência de dados elaborado neste trabalho, cujos objetivos foram levantados no capítulo anterior.

3.1.
Esquema Conceitual de Proveniência

Foi construído um esquema (Figura 16) de dados de proveniência usando modelagem entidade-relacionamento. O esquema pode ser dividido em três partes: *Atividades*, *Definição do Workflow*, e *Execução do Workflow*.

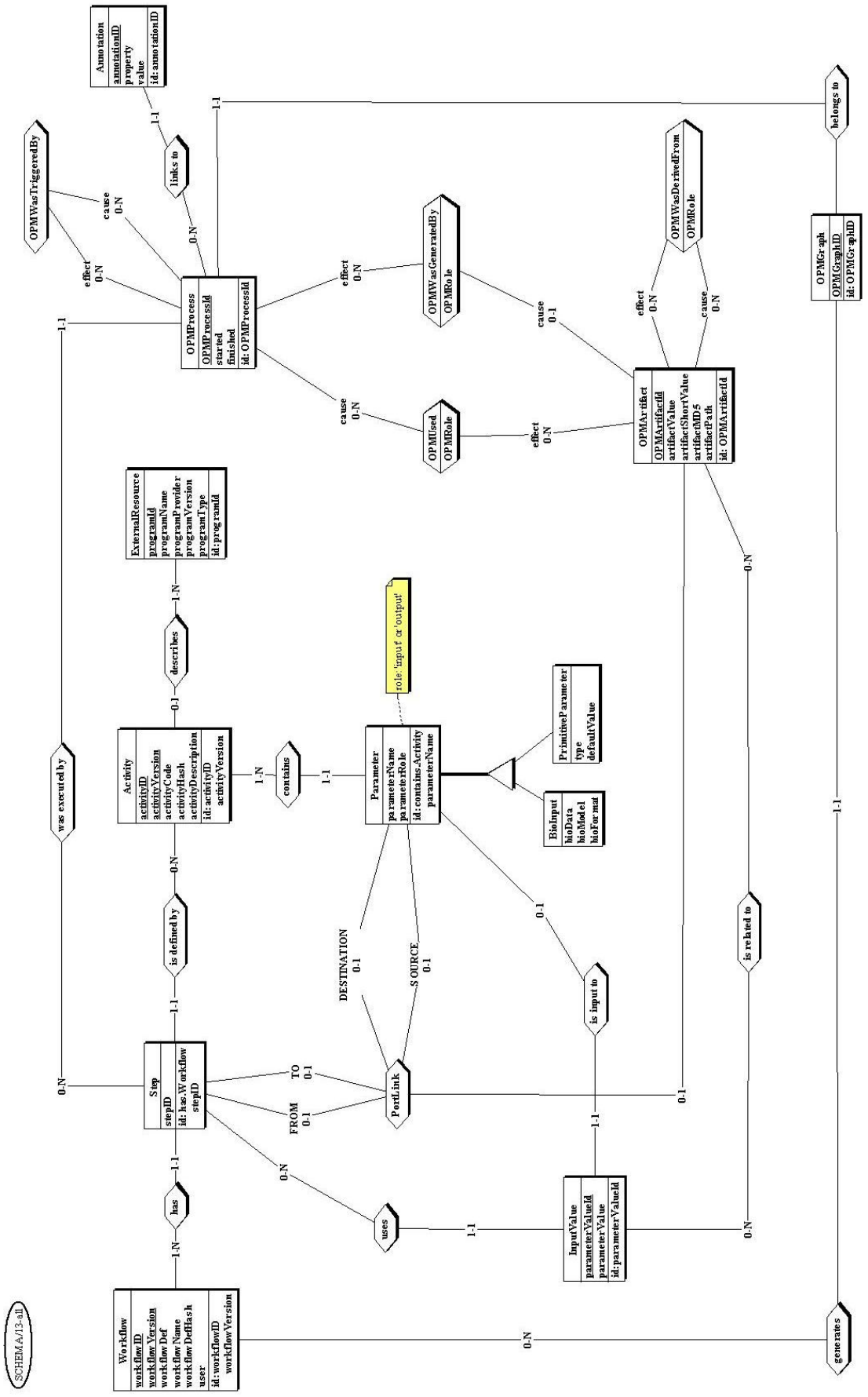

Figura 16 – Esquema ER de proveniência de dados

3.1.1. Atividades

O esquema proposto (Figura 17) contém 5 entidades que suportam a representação de descrição de atividades disponíveis no sistema:

- Entidade *Activity*: Contém o identificador único da atividade, a versão, uma descrição textual, o próprio arquivo da atividade (activityCode) e um atributo para armazenar o MD5 do mesmo. Note-se que o arquivo pode ser um xml no caso do BioSide e futuramente do Taverna, um script Python (Vistrails), uma classe Java (Kepler).
- Entidade *Parameter*: Representa os parâmetros da atividade. Um parâmetro tem um nome, e pode representar um dado de entrada ou saída, informação dada pelo atributo *role*.
- Entidade *BioInput*: Especialização da entidade parâmetro. Visa representar arquivos de dados biológicos que são esperados pelas atividades. Possuem um tipo de dado biológico (ex.: DNA, proteína, RNA), um modelo (ex.: alinhamento, árvore filogenética, matriz de similaridade), e um formato de arquivo esperado (ex.: FASTA, newick).
- Entidade *PrimitiveParameter*: Especialização da entidade Parameter. Visa representar parâmetros de tipos primitivos. Possuem um tipo (ex.: string, double) e um valor padrão.
- Entidade *ExternalResource*: Representa um recurso externo acessado pela atividade. Possui como atributos um identificador, o nome do recurso, a instituição provedora, a versão e o tipo (ex.: alinhamento de sequências, clusterização).
- Relacionamento *describes*: Se a atividade descreve algum recurso externo haverá um relacionamento entre a mesma e o recurso, descrito pela entidade ExternalResource. Uma atividade pode descrever apenas um recurso, e um recurso pode ser descrito por várias atividades.
- Relacionamento *contains*: Representa a ligação entre a atividade e seus parâmetros. Uma atividade contém vários parâmetros, e um parâmetro pertence a uma única atividade.

A proveniência de atividades é importante para a compreensão do workflow, para a reprodutibilidade no nível de reuso de definição, e para o compartilhamento da descrição da atividade.

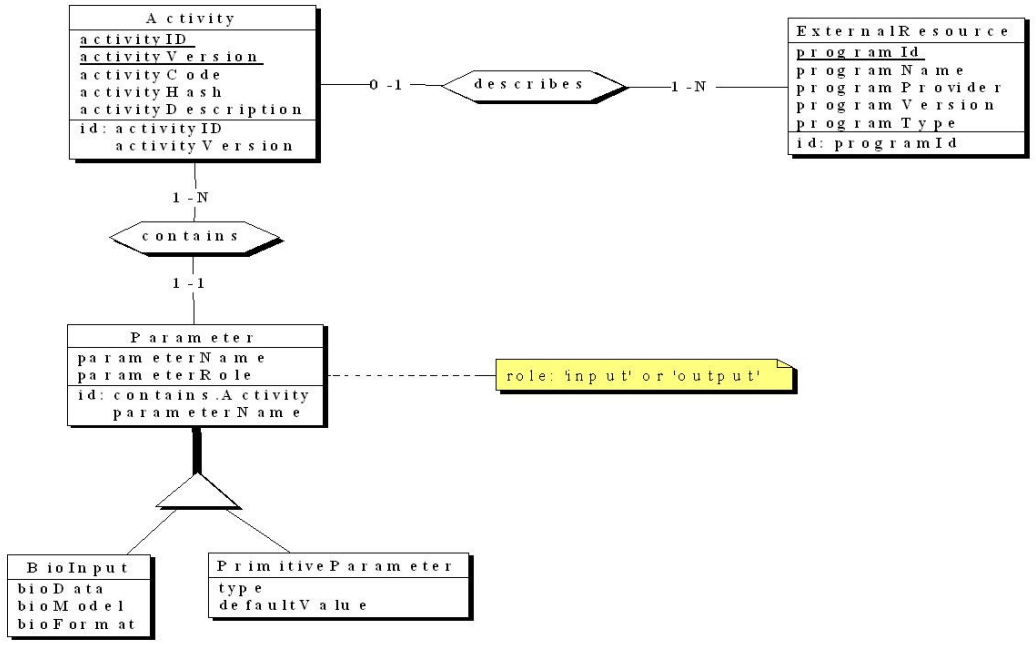

Figura 17 – Descrição de Atividades

3.1.2. Definição do Workflow

Na parte do modelo que representa a definição (Figura 18) do workflow temos as entidades Workflow, Step, InputValue. Um workflow consiste em um conjunto de passos (entidade 'Step'), que se referem às atividades disponíveis no sistema.

- Entidade *Workflow*: Representa o workflow, que possui um identificador único, uma versão, um nome, o usuário que o criou, a própria definição do mesmo (um arquivo cujo formato depende do sistema), e um atributo para registrar o MD5 da mesma.
- Entidade *Step*: Representa o *passo* do workflow, tendo apenas um identificador utilizado para diferenciar passos do workflow que se referem a uma mesma atividade.
- Entidade *InputValue*: Representa o valor de uma entrada de dados definida pelo usuário. Este valor pode ser um caminho de arquivo ou um valor primitivo. No caso em que o usuário deseja reutilizar

um artefato, o InputValue terá uma referência para um artefato, que estará representada no relacionamento *is related to*.

- Relacionamento *has*: Um workflow tem vários passos, e um passo é referente a uma única definição de workflow.
- Relacionamento *PortLink*: Define as dependências entre os passos, que são visualmente as setas do workflow. Uma ligação entre dois passos possui um passo de origem e um passo de destino, mas apenas esta informação não basta para especificar a ligação, visto que as atividades possuem vários parâmetros que podem participar do workflow como portas de entrada e saída. Os relacionamentos *Source* e *Destination* com a entidade Parameter informam qual a porta de origem e destino da ligação.
- Relacionamento *is defined by*: Um passo tem relação com uma única atividade, e uma atividade pode ser referenciada por vários passos.
- Relacionamento *uses*: Um passo utiliza valores definidos pelo usuário para seus parâmetros. Vários parâmetros podem ser instanciados pelo usuário, e um valor de parâmetro tem relação com apenas um passo.
- Relacionamento *is input to*: Um valor de parâmetro é definido para um parâmetro específico, e um parâmetro pode ter ou não um valor informado pelo usuário.
- Relacionamento *is related to*: Um InputValue pode estar relacionado a vários artefatos, no caso de várias execuções de uma mesma definição de workflow. Um artefato pode ser referenciado por várias instâncias de InputValue, no caso de reuso de artefatos.

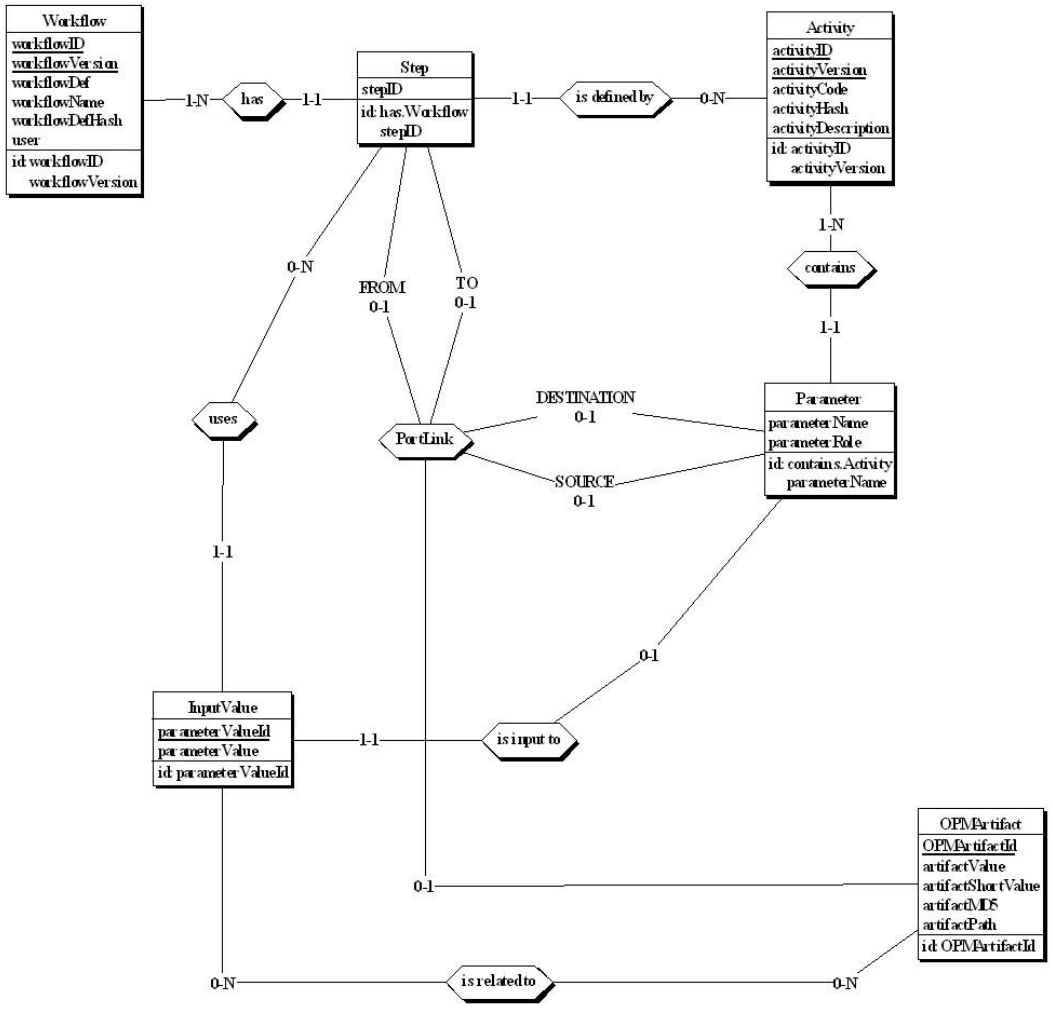

Figura 18 – Proveniência prospectiva

3.1.3.
Execução do Workflow

A parte retrospectiva do esquema (Figura 19) foi elaborada com base em algumas entidades definidas pelo OPM. Para visualizar os relacionamentos com a parte de definição vide figura completa (Figura 16).

- Entidade *OPMGraph*: Representa uma execução de workflow. Uma definição de workflow pode estar relacionada a várias execuções, e uma execução a apenas uma definição (relacionamento *generates*).
- Entidade *OPMArtifact*: Representa um dado produzido em uma execução, que poderá ser um valor primitivo ou um arquivo. Se o dado foi produzido no fluxo, a ligação com o relacionamento *PortLink* indica em qual porta o arquivo foi produzido e por qual porta foi consumido. Para casos em que o dado foi informado pelo

usuário, há a ligação com a entidade *InputValue* da parte de definição. O atributo artifactValue é utilizado para armazenar valores de parâmetros primitivos (ex.: string, boolean). Este valor já se encontra na definição, mas também é armazenado em OPMArtifact para suportar os casos em que os valores podem ser alterados em tempo de execução. No caso de arquivos, o atributo artifactShortValue é usado para armazenar a parte inicial do arquivo, artifactPath registra o caminho completo do mesmo após ser copiado para o diretório de execução, e artifactMD5 o código MD5 do arquivo.

- Entidade *OPMProcess*: Representa os dados de execução de um passo do workflow. Possui apenas o horário de início e fim do passo, uma vez que outras anotações são registradas na entidade Annotation. A indicação do passo referente é realizada via relacionamento *was executed by*. Um OPMProcess se refere a apenas um passo, e um passo pode ter relação com vários processos. O relacionamento *belongs to* com a entidade OPMGraph indica a que execução se refere o processo.
- Entidade *Annotation*: Utilizada para registrar anotações sobre a execução dos processos, como mensagens de erro e códigos de saída, ou anotações sobre artefatos. Cada anotação possui o nome de uma propriedade e seu valor, seguindo o direcionamento do OPM para inclusão de anotações. Algumas anotações que podemos armazenar:
    - Valor de saída do processo (exitValue)
    - Mensagens de erro da saída padrão (stdOut)
    - MD5 do arquivo executável invocado
- Relacionamentos *OPMWasGeneratedBy* e *OPMUsed*: São utilizados para registrar qual processo gerou um artefato e quais artefatos foram utilizados por um determinado processo.
- Relacionamentos *OPMWasTriggeredBy* e *OPMWasDerivedFrom*: São utilizados para registrar relações de dependências entre processos e entre dados, respectivamente. Estas relações não são gravadas diretamente ao receber um resultado de execução, mas podem ser inferidas de acordo com as regras de inferência do OPM.

Optamos por não incluir no esquema a entidade *Account* e a entidade *Agent*, uma vez que seu uso é opcional e não interfere nos objetivos do trabalho realizado.

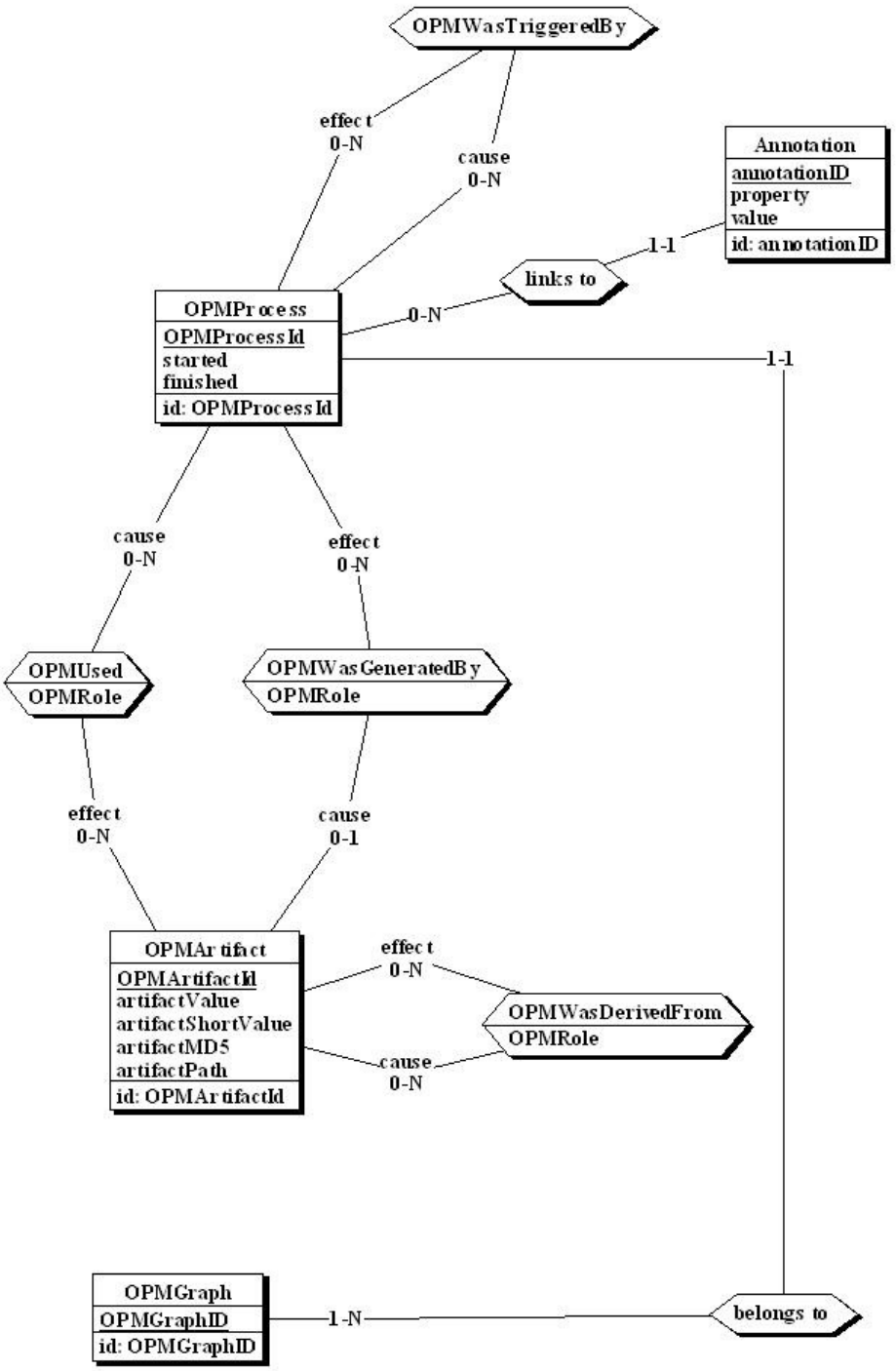

Figura 19 – Proveniência retrospectiva

3.1.4.
Discussão sobre Parâmetros e Portas

Um passo de workflow deve ser parametrizado pelo usuário para que sejam definidos valores de dados de entrada e de variáveis de configuração de execução. Os parâmetros das atividades podem ser chamados também de *portas*, uma vez que estabelecem a forma como a atividade recebe dados de entrada ou produz dados de saída.

Alguns sistemas (BioSide, Kepler, REDUX) consideram portas e parâmetros duas entidades distintas. As portas são utilizadas para o fluxo de dados no workflow, e os parâmetros são configurados manualmente pelo usuário. No Kepler existe ainda uma entidade PortParameter. Este tipo de parâmetro pode ser configurado pelo usuário na interface de parametrização, ou pode ser usado como porta para entrada de dados produzidos por outro ator. Se for usado como porta, o valor que é recebido sobrescreve a configuração feita pelo usuário caso exista.

Já nos sistemas Vistrails e e-bioflow todos os parâmetros são chamados de portas, que podem ser instanciadas por outra atividade formando o fluxo, ou pelo próprio usuário.

Os programas de Bioinformática normalmente possuem parâmetros que representam arquivos biológicos, como bancos de dados, listas de sequências biológicas, alinhamentos entre sequências e outros tipos de arquivos do domínio em questão. Estes parâmetros são candidatos imediatos para a participação no fluxo de dados, pois são dados que podem ser produzidos por outro programa de Bioinformática. Outros tipos de parâmetros são os que normalmente são definidos manualmente pelo usuário, por exemplo, o *e-value* do BLAST, que é uma medida de probabilidade que afeta o comportamento do programa. Esses parâmetros em geral não participarão do fluxo de dados principal do workflow, pois são configurações normalmente estipuladas pelo usuário, que podem assumir tipos de dados primitivos (*strings*, *double*, *boolean* etc), ou podem ser também arquivos.

Para a melhor representação de tarefas de Bioinformática, optamos por representar no esquema de dados duas especializações da entidade Parameter, BioInput e PrimitiveParameter. Estas especializações permitem o melhor entendimento do tipo do parâmetro e facilitam a verificação de compatibilidade entre a saída de um programa e a entrada de outro.

Embora o fluxo de dados seja estabelecido em geral utilizando parâmetros que são arquivos, optamos por não restringir o fluxo de dados apenas aos parâmetros do tipo BioFile. Desta forma, qualquer parâmetro pode ser utilizado como uma porta de entrada ou saída, sem criar a limitação na qual apenas arquivos podem participar do fluxo de dados, como acontece na modelagem original do BioSide.

3.2. Suporte aos Desafios

Serão apresentadas nesta seção as contribuições do esquema proposto para atingir os objetivos elicitados na seção 2.3. Implementamos o esquema de dados utilizando modelagem relacional a fim de demonstrar com exemplos práticos essas contribuições. No anexo 1 estão descritas as tabelas do banco de dados de proveniência.

3.2.1. Reprodutibilidade

No esquema proposto, o banco de dados não sofrerá alterações, mas apenas inserções. Optamos por essa estratégia para atender o requisito de reprodutibilidade. A especificação de um workflow é descrita por um arquivo, que é gravado no campo workflowDef da tabela Workflow. É registrado também o código MD5 desse arquivo no campo workflowDefHash. Uma definição de workflow que tenha sido alterada deve ser regravada no banco como uma nova versão dessa definição, registrada no campo workflowVersion. Assim também ocorre com as atividades utilizadas. Esta estratégia de versionamento mantém imutáveis os dados de proveniência em relação à definição referente a uma execução e em relação às atividades utilizadas em uma definição de workflow.

A ligação entre a definição e a execução possibilita a reprodução de uma execução de um workflow usando a mesma definição e as mesmas atividades usadas no momento da execução. Dada uma execução de workflow a definição relativa a esta execução é mantida intacta para que seja possível que a partir de um grafo OPM se saiba qual era a definição a ser executada. Note-se que não necessariamente tudo o que está definido foi executado. Por exemplo, um componente pode ter falhado e assim todos os subsequentes também. A ligação do grafo de execução com a parte de definição do workflow permite ao usuário

consultar qual era o workflow pensado para execução, quais componentes deveriam ser executados e não foram, por exemplo.

Em relação à reprodução de acessos externos, na parte prospectiva do modelo temos algumas informações sobre o programa invocado na tabela ExternalResource. Já na parte retrospectiva pode ser incluído como anotação de processos (entidade Annotation) o caminho completo do programa executado. É possível que na descrição da atividade esteja especificado apenas o nome do executável, o que é suficiente se o mesmo está disponível no PATH do sistema. Capturar o caminho completo é importante porque é possível que existam dois aplicativos diferentes porém de mesmo nome instalados em locais diferentes no PATH. O sistema operacional irá escolher qual aplicativo executar, e a escolha pode ser diferente da esperada pelo usuário. Ao receber os dados de proveniência da execução, o sistema irá também calcular o MD5 do aplicativo invocado e registrá-lo como anotação. Este dado permite ao usuário saber se um aplicativo disponível em sua configuração é o mesmo que foi executado.

Estes registros oferecem mais informações de proveniência para a identificação e reprodução de acessos externos, em comparação com outros SGWC estudados, como Vistrails, Taverna e Kepler. Porém é importante ressaltar que ainda que este seja um passo a mais na direção da reprodutibilidade estrita, ainda não é possível garantir a mesma, visto que não estão sendo consideradas as dependências de bibliotecas e outros recursos, que podem ter sido alterados sem o conhecimento do sistema.

A reprodutibilidade depende também do acesso aos dados originais, ou metadados, consumidos e produzidos em uma execução. Discutiremos esse aspecto na seção a seguir.

3.2.2. Gerência de Dados Consumidos e Produzidos

Todos os arquivos e dados gerados e consumidos em uma execução terão metadados armazenados na tabela OPMArtifact. Um dado primitivo (double, boolean, string) será apenas regravado no campo artifactValue. Optamos por incluir o valor do dado na tabela Artifact (além da tabela InputValue, da definição do workflow) para suportar casos em que o sistema permite a alteração do dado em tempo de execução.

O esquema proposto permite que a partir de um artefato se possa identificar qual a execução que o gerou, qual a definição de workflow que foi

executada e com que artefatos possui uma relação de causalidade. A Figura 20 ilustra essas ligações de proveniência.

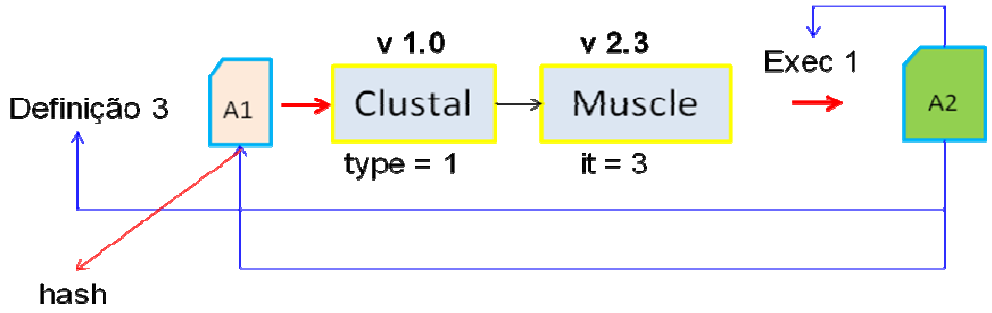

Figura 20 – Ligações de proveniência

Em relação aos arquivos, o atributo artifactShortValue permite a gravação de uma parte inicial do arquivo para que o usuário possa visualizar uma prévia do mesmo. Já o atributo artifactMD5 registra o cálculo MD5 sobre o conteúdo do arquivo.

Para garantir a reprodutibilidade estrita de uma execução, não basta registrar referências para os arquivos e mantê-los em diretórios usuais, uma vez que os arquivos podem ser inadvertidamente alterados pelo usuário, movidos ou apagados. Aqui propomos que os arquivos sejam copiados para um diretório específico do sistema de gerência, o que promove maior controle dos mesmos.

O arquivo será gerado em uma pasta de nome construído a partir da identificação da definição do workflow e de sua execução específica. O *path* do arquivo deve ser então registrado no campo artifactPath.

Esta gravação de uma cópia do arquivo original deve ficar à escolha do pesquisador, uma vez que nem sempre será necessária a reprodução estrita das execuções. Por isso o SGWC deve prover ao usuário uma opção de habilitar ou desabilitar a cópia de arquivos. O pesquisador pode efetuar diversos testes de seu workflow com essa opção desabilitada a fim de não perder tempo nem espaço de armazenamento. Ao realizar execuções que serão publicadas em artigos ou relatórios técnicos a opção de gravação poderia ser habilitada a fim de permitir a reprodução estrita dessas execuções, gerando confiabilidade de suas publicações.

Nos casos em que o usuário mantém desabilitada a opção de cópia de arquivos, mesmo não tendo uma cópia do arquivo original, o esquema proposto permite a gravação do código MD5. Isso permite que o usuário identifique se um

arquivo que ele tem disponível é o mesmo que foi utilizado na execução, bastando calcular seu código MD5 e verificar se é igual ao registrado no sistema.

3.2.3. Reuso de Dados

Todos os artefatos produzidos pelas execuções de workflows possuem no esquema proposto um relacionamento com o seu processo gerador (OPMProcess), através do relacionamento OPMWasGeneratedBy. Um OPMProcess por sua vez possui uma ligação com um Step de uma definição específica. Dessa forma, é possível verificar qual a origem de um artefato, ou seja, a execução que o gerou junto à correspondente definição de workflow.

Para promover o uso de um artefato produzido em alguma execução de workflow como entrada em outro workflow foi incluído no esquema proposto o relacionamento *is related to*, doravante mapeado para a tabela InputValue_Artifact. Este relacionamento permite que um dado de entrada para um determinado passo de workflow seja um artefato produzido em outro workflow, tal como elicitado no desafio em questão.

Note-se que a solução proposta resolve o problema no nível de modelagem conceitual apenas. Para que o usuário possa de fato reutilizar um artefato, o SGWC deve ser alterado para prover uma funcionalidade de busca e escolha de artefatos na fase de composição do workflow.

3.2.4. Descrição e Gerência de Atividades

O esquema proposto neste trabalho inclui descrições para as atividades. O próprio arquivo de definição da atividade deve ser armazenado no campo activityCode da tabela Activity. Este registro é importante para garantir a compreensão e reprodutibilidade de um experimento caso não se tenha mais acesso ao código da atividade via SGWC, ou se o pesquisador desejar compartilhar seu experimento com um usuário que não possua a atividade instalada.

O código MD5 do arquivo descritor da atividade será armazenado no campo activityHash. Para exemplificar como pode ser realizado o versionamento das atividades, consideremos duas execuções (Figura 21) de um workflow que utiliza a atividade BATS, que é descrita por um script em Perl. Suponha que na primeira execução a atividade BATS utilizada seja encontrada no banco. Para

verificar se a atividade já está armazenada basta calcular o código MD5 da mesma e procurá-lo no banco de dados. Ao registrar a segunda execução, caso a atividade BATS utilizada não seja mais encontrada no banco de dados ela será armazenada como um novo registro e receberá uma nova versão de forma incremental. Esta estratégia de versionamento permite que uma atividade que foi utilizada em uma execução possua um registro que permanece intacto, o que é importante para a reprodutibilidade estrita e não é considerado por alguns sistemas. O Vistrails e o Kepler armazenam apenas uma referência para a atividade. Já o Taverna armazena o código da atividade todas as vezes que o workflow é executado, sem realizar nenhum tipo de versionamento.

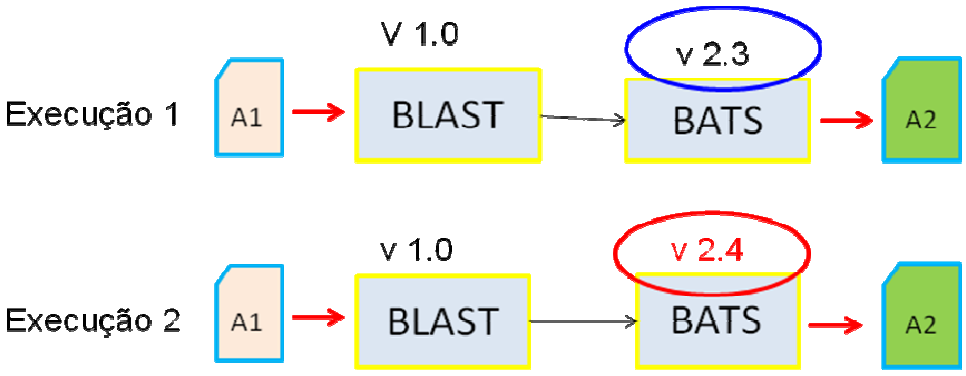

Figura 21 – Versionamento de atividades

Com relação a atividades que acessam recursos externos, representamos também na entidade *ExternalResource* alguns metadados sobre o programa acessado. Este registro não é realizado em nenhum dos sistemas estudados, e se torna especialmente importante para workflows de BioInformática devido ao intenso uso de programas stand-alone.

Os parâmetros podem ser especializados em duas entidades, BioInput e PrimitiveParameter. A entidade BioInput permite o registro de dados biológicos sobre o arquivo esperado pela atividade. Estes dados facilitam ao usuário o entendimento da atividade e a construção de workflows válidos, pois permitem que o sistema valide se as passagens de dados estão nos formatos esperados.

3.2.5. Modelo de Proveniência

Com relação a modelagem de proveniência, buscamos uma solução conceitual a fim de manter distanciamento de detalhes específicos da tecnologia utilizada em cada SGWC. O esquema proposto não esgota todas as informações

possíveis em relação a definição de workflows, o que representaria uma nova linguagem de definição. O interesse deste trabalho foi extrair algumas informações relevantes da definição do workflow, suas atividades e suas execuções de forma a armazenar a proveniência e permitir consultas.

Com relação ao uso do OPM, embora este modelo tenha surgido como proposta de padronização para representação em alto nível de dados de proveniência, identificamos que poucos sistemas o têm utilizado diretamente em seus modelos. Isto se deve parcialmente ao fato de que o OPM suporta apenas representação de proveniência retrospectiva.

Para aproveitar toda a formalização já obtida no OPM faz-se necessário criar um modelo de proveniência que o utilize no que ele oferece, sendo complementado com a modelagem da proveniência prospectiva. Este objetivo foi alcançado no esquema proposto através da ligação entre a proveniência retrospectiva representada utilizando OPM e a parte da proveniência prospectiva.

3.3. Conclusão

Neste capítulo foi apresentado o projeto de representação de dados de proveniência para SGWC desenvolvido neste trabalho.

Alguns objetivos principais do projeto foram atingidos embora de forma limitada. O modelo de proveniência é independente da tecnologia utilizada em um sistema específico, mas tem falhas de generalidade por não tratar fluxos de controle e acesso a web services.

Embora não tenha sido utilizada nenhuma abordagem específica de gerenciamento de configuração para as atividades, elaboramos uma proposta que contribui para a proveniência das atividades utilizadas nos workflows. Essa proposta poderia ser aprimorada através do uso de uma ferramenta de gerência de configuração. O mesmo se pode dizer do mecanismo de proveniência proposto para o versionamento de definições de workflows, que também poderia ser tratado sob ponto de vista de gerência de configuração.

Foi elaborada uma proposta de descrição de atividades que considera dois níveis de descrição, o da atividade e o do recurso externo. Além disso propomos que os parâmetros devem ser especializados em tipos de dados de acordo com o domínio do problema, neste caso o domínio da Bioinformática.

Podemos listar como limitações do projeto a falta de suporte ao uso de subworkflows, a falta de modelagem de atividades em nível abstrato (seção 2.2.2.8) e falta de representação de um experimento científico como uma possível composição de vários workflows.