UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gabriella Castro Barbosa Costa

Uma Abordagem para Linha de Produtos de Software Científico Baseada em Ontologia e *Workflow*

Juiz de Fora

2013

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gabriella Castro Barbosa Costa

Uma Abordagem para Linha de Produtos de Software Científico Baseada em Ontologia e *Workflow*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Regina Maria Maciel Braga Villela
Coorientador: José Maria Nazar David

Juiz de Fora
2013

Ficha catalográfica elaborada através do Programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Costa, Gabriella Castro Barbosa.
Uma Abordagem para Linha de Produtos de Software Científico Baseada em Ontologia e Workflow / Gabriella Castro Barbosa Costa. -- 2013.
103 f. : il.

Orientadora: Regina Maria Maciel Braga Villela
Coorientador: José Maria Nazar David
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2013.

1. Linha de Produtos. 2. Ontologia. 3. Modelo de Features. 4. Workflow Científico. 5. Alinhamento de Sequências. I. Villela, Regina Maria Maciel Braga, orient. II. David, José Maria Nazar, coorient. III. Título.

Gabriella Castro Barbosa Costa

Uma Abordagem para Linha de Produtos de Software Científico Baseada em Ontologia e *Workflow*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 27 de Fevereiro de 2013.

BANCA EXAMINADORA

Profa. D.Sc. Regina Maria Maciel Braga Villela - Orientadora
Universidade Federal de Juiz de Fora

Prof. D.Sc. José Maria Nazar David - Coorientador
Universidade Federal de Juiz de Fora

Profa. D.Sc. Claudia Maria Lima Werner
Universidade Federal do Rio de Janeiro

Profa. D.Sc. Fernanda Cláudia Alves Campos
Universidade Federal de Juiz de Fora

Aos meus pais e ao meu noivo.

AGRADECIMENTOS

A Deus, por estar sempre ao meu lado em todos os momentos.

Aos meus pais, Heloiza e José Wilson, por sempre acreditarem em meus sonhos.

Ao meu noivo, Humberto, pela paciência e companheirismo de todos os dias, mesmo nos momentos mais difíceis.

Aos meus orientadores, Prof.ª Regina Braga e Prof. José Maria, pela dedicação, paciência, incentivo e grande auxílio no decorrer de todo o curso de Mestrado.

Aos membros da Banca Examinadora pelo trabalho de avaliação.

Ao Wagner Arbex e à Kele Belloze pela disponibilidade e auxílio que muito contribuíram para a realização deste trabalho.

Aos alunos do NEnC, em especial ao Magnus Ribeiro, pelas dicas e contribuições durante a etapa de implementação.

À Gláucia Vargas, pela disponibilidade e paciência em atender a todas as minhas solicitações.

A todos os professores e colegas que tive durante o Mestrado, por todos os ensinamentos.

À Universidade Federal de Juiz de Fora, pelo apoio financeiro através da bolsa.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

"Sucesso e genialidade são 10 por cento de inspiração e 90 por cento de transpiração"

Albert Einstein

RESUMO

Uma forma de aprimorar a reutilização e a manutenção de uma família de produtos de software é através da utilização de uma abordagem de Linha de Produtos de Software (LPS). Em algumas situações, tais como aplicações científicas para uma determinada área, é vantajoso desenvolver uma coleção de produtos de software relacionados, utilizando uma abordagem de LPS. Linhas de Produtos de Software Científico (LPSC) diferem-se de Linhas de Produtos de Software pelo fato de que LPSC fazem uso de um modelo abstrato de *workflow* científico. Esse modelo abstrato de *workflow* é definido de acordo com o domínio científico e, através deste *workflow*, os produtos da LPSC serão instanciados. Analisando as dificuldades em especificar experimentos científicos e considerando a necessidade de composição de aplicações científicas para a sua implementação, constata-se a necessidade de um suporte semântico mais adequado para a fase de análise de domínio. Para tanto, este trabalho propõe uma abordagem baseada na associação de modelo de *features* e ontologias, denominada PL-Science, para apoiar a especificação e a condução de experimentos científicos. A abordagem PL-Science, que considera o contexto de LPSC, visa auxiliar os cientistas através de um *workflow* que engloba as aplicações científicas de um dado experimento. Usando os conceitos de LPS, os cientistas podem reutilizar modelos que especificam a LPSC e tomar decisões de acordo com suas necessidades. Este trabalho enfatiza o uso de ontologias para facilitar o processo de aplicação de LPS em domínios científicos. Através do uso de ontologia como um modelo de domínio consegue-se fornecer informações adicionais, bem como adicionar mais semântica ao contexto de LPSC.

Palavras-chave: Linha de Produtos. Ontologia. Modelo de *Features*. *Workflow* Científico. Alinhamento de Sequências.

ABSTRACT

A way to improve reusability and maintainability of a family of software products is through the Software Product Line (SPL) approach. In some situations, such as scientific applications for a given area, it is advantageous to develop a collection of related software products, using an SPL approach. Scientific Software Product Lines (SSPL) differs from the Software Product Lines due to the fact that SSPL uses an abstract scientific workflow model. This workflow is defined according to the scientific domain and, using this abstract workflow model, the products will be instantiated. Analyzing the difficulties to specify scientific experiments, and considering the need for scientific applications composition for its implementation, an appropriated semantic support for the domain analysis phase is necessary. Therefore, this work proposes an approach based on the combination of feature models and ontologies, named PL-Science, to support the specification and conduction of scientific experiments. The PL-Science approach, which considers the context of SPL and aims to assist scientists to define a scientific experiment, specifying a workflow that encompasses scientific applications of a given experiment, is presented during this dissertation. Using SPL concepts, scientists can reuse models that specify the scientific product line and carefully make decisions according to their needs. This work also focuses on the use of ontologies to facilitate the process of applying Software Product Line to scientific domains. Through the use of ontology as a domain model, we can provide additional information as well as add more semantics in the context of Scientific Software Product Lines.

Keywords: Product Line. Ontology. Feature Model. Scientific Workflows. Sequence Alignment.

LISTA DE FIGURAS

2.1 Exemplo de Modelo de *Features* 23
2.2 Representação de Restrições no Modelo de *Features* 25
2.3 Modelo Definido (ou Declarado) 28
2.4 Modelo Inferido 29
2.5 Computação e o Terceiro Pilar [adaptado de (RIEDEL et al., 2008)]. 30

3.1 Modelo Abstrato de *Workflow* Científico 35
3.2 Plataformas de Sequenciamento 36
3.3 Tarefas para Sequenciamento/Alinhamento Genético 37
3.4 Restrições da Classe *BLASTN* 39
3.5 Ontologia *myGrid:* Classes Principais 39
3.6 *Sequence Alingment Ontology:* Classes Principais 40
3.7 Ontologia *myGrid:* Classe *bioinformatics_task* 41
3.8 *Sequence Alignment Ontology:* Classe *bioinformatics_task* 41
3.9 Ontologia *myGrid:* Restrições da Classe *pairwise_local_aligning* 42
3.10 *Sequence Alignment Ontology:* Restrições da Classe *pairwise_local_aligning* . . 42
3.11 Função do Arquivo de Mapeamento Especificado em XML 43
3.12 Trecho do Arquivo de Mapeamento 44
3.13 Modelo Abstrato de *Workflow* para Sequenciamento/Alinhamento Genético . 44
3.14 Visão Geral da Arquitetura PL-Science 45
3.15 Artefatos Manipulados pelo Gerente de Variabilidade 46
3.16 Funções iniciais do Gerente da Linha de Produtos 47
3.17 Tela inicial da Aplicação 48

4.1 Modelo Abstrato de *Workflow* para Sequenciamento/Alinhamento Genético . 58
4.2 Tipos de Alinhamento 61
4.3 Modelo Abstrato de *Workflow* Criado 63
4.4 Tarefas do Modelo Abstrato de *Workflow* 64
4.5 Restrições da Classe *sequence_grouping* 64
4.6 Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 1) 66

4.7 Exemplo 1 - Tela 1 67
4.8 Classe correspondente à *feature workflow* 67
4.9 Restrições da Classe *isolated_application* 68
4.10 Restrições da Classe *workflow* 68
4.11 Exemplo 1 - Tela 2 68
4.12 Classe correspondente à *feature Sanger* 69
4.13 Restrições da Classe *Sanger* 69
4.14 Exemplo 1 - Tela 3 70
4.15 Exemplo 1 - Seleção da última tarefa do *workflow* a ser criado 71
4.16 Indivíduo *runPhrapService* 72
4.17 Exemplo de Inferência sobre a Classe *PHRED* 73
4.18 Exemplo de Inferência sobre a Classe *Cross_match* 73
4.19 Exemplo de Inferência sobre a Classe *PHRAP* 73
4.20 Arquivo XML gerado pela Aplicação (Produto 1) 74
4.21 Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 2) 75
4.22 Exemplo 2 - Tela 1 76
4.23 Classe correspondente à *feature isolated_application* 76
4.24 Restrições da Classe *isolated_application* 76
4.25 Exemplo 2 - Tela 2 77
4.26 Classe correspondente à *feature NGS* 77
4.27 Restrições da Classe *NGS* 78
4.28 Exemplo 2 - Tela 3 79
4.29 Exemplo 2 - Seleção da tarefa desejada 80
4.30 Exemplo de Inferência sobre a Classe *ClustalW2* 80
4.31 Sequências de entrada suportadas pelo *Clustal_Omega* 81
4.32 Sequências de entrada suportadas pelo *ClustalW2* 81
4.33 Arquivo XML gerado pela Aplicação (Produto 2) 82
4.34 Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 3) 83
4.35 Exemplo 3 - Tela 1 83
4.36 Classe correspondente à *feature workflow* 84
4.37 Restrições da Classe *workflow* 84
4.38 Exemplo 3 - Tela 2 85

4.39 Classe correspondente à *feature Sanger* 85
4.40 Restrições da Classe *Sanger* 86
4.41 Exemplo 3 - Tela 3 86
4.42 Seleção da *feature* que representa a última tarefa do *workflow* 87
4.43 Exemplo de Inferência sobre a Classe *PHRED* 87
4.44 Exemplo de Inferência sobre a Classe *Cross_match* 88
4.45 Exemplo de Inferência sobre a Classe *PHRAP* 88
4.46 Detalhamento da tarefa de Alinhamento 89
4.47 Inferência sobre a Classe *BLAST* 89
4.48 Arquivo XML gerado pela Aplicação (Produto 3) 90

LISTA DE TABELAS

3.1 Comparação dos Trabalhos Relacionados 54

SUMÁRIO

1 INTRODUÇÃO 14

1.1 MOTIVAÇÃO 14

1.2 JUSTIFICATIVA 14

1.3 OBJETIVOS 15

1.4 ESTRUTURA DO TRABALHO 16

2 PRESSUPOSTOS TEÓRICOS 17

2.1 LINHA DE PRODUTOS DE SOFTWARE (LPS) 17

2.1.1 Desenvolvimento de uma LPS 18

2.1.1.1 Desenvolvimento do Núcleo de Artefatos 19

2.1.1.2 Desenvolvimento do Produto usando o Núcleo de Artefatos 20

2.2 VARIABILIDADE E MODELOS DE *FEATURES* 21

2.3 ONTOLOGIAS 26

2.3.1 Inferência 28

2.4 SOFTWARE CIENTíFICO 30

2.5 CONSIDERAÇÕES FINAIS DO CAPíTULO 33

3 A ABORDAGEM PL-SCIENCE 34

3.1 INTRODUÇÃO 34

3.2 MODELOS DE DOMíNIO NO CONTEXTO DA PL-SCIENCE 35

3.2.1 Modelo de *Features* 36

3.2.2 Ontologia 38

3.2.3 Mapeamentos 42

3.2.4 Modelo Abstrato de *Workflow* Científico 44

3.3 ARQUITETURA DA SOLUÇÃO 45

3.3.1 Gerente de Variabilidade (*Variability Manager*) 46

3.3.2 Gerente da Linha de Produtos (*Product Line Manager*) 47

3.3.3 Gerente Arquitetural (*Architectural Manager*) 49

3.4 METODOLOGIA 49

3.5 TRABALHOS RELACIONADOS 51

3.6 CONSIDERAÇÕES FINAIS DO CAPíTULO 55

4 IMPLEMENTAÇÃO DA ABORDAGEM PL-SCIENCE 56

4.1 INTRODUÇÃO 56

4.2 DOMíNIO DE SEQUENCIAMENTO/ALINHAMENTO GENÉTICO 57

4.2.1 Sequenciamento 57

4.2.2 Alinhamento 58

4.2.3 Modelo Abstrato de *Workflow* para Sequenciamento/Alinhamento Genético. 58

4.3 APLICAÇÃO DA ABORDAGEM PL-SCIENCE 62

4.3.1 Produto 1 - Pipeline PhredPhrap 65

4.3.2 Produto 2 - Tarefa isolada para Alinhamento Múltiplo de Sequências 74

4.3.3 Produto 3 - *Workflow* para Sequenciamento e Alinhamento 82

4.4 CONSIDERAÇÕES FINAIS DO CAPíTULO 90

5 CONCLUSÕES 93

REFERÊNCIAS 95

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Através da utilização de uma abordagem de desenvolvimento usando Linha de Produtos de Software (LPS) é possível aprimorar a reutilização e facilitar a manutenção de uma família de produtos de software. Famílias de Software, como também podem ser chamadas as LPS, empregam amplamente o conceito de reutilização por serem definidas como um conjunto de sistemas que são desenvolvidos a partir de um mesmo conjunto de artefatos base (CLEMENTS; NORTHROP, 2001).

Em algumas situações, como é o caso de aplicações científicas para uma determinada área, é vantajoso desenvolver uma coleção de produtos de software relacionados, utilizando uma abordagem de LPS. Se os produtos de software são semelhantes o suficiente, então há a possibilidade de prever o que eles têm em comum, o que difere entre eles e, em seguida, reutilizar estes aspectos comuns em prol do desenvolvimento de novas aplicações no meio científico.

1.2 JUSTIFICATIVA

A maioria dos usuários de *frameworks* científicos são cientistas em um campo específico de pesquisa, os quais não possuem, necessariamente, formação adequada para desenvolvimento de software. Muitas vezes eles iniciam uma nova aplicação copiando uma já existente e simplesmente adaptam a mesma às suas necessidades. A Engenharia de Linha de Produtos pode ajudar a entender melhor sobre o software que está sendo desenvolvido, pois os cientistas podem seguir um modelo que especifica a linha de produtos e cuidadosamente tomar as decisões, de acordo com suas necessidades, para cada ponto de variação[1] deste modelo (REMMEL et al., 2011). Portanto, tendo em vista que o uso de LPS traz como vantagem o reúso sistemático, esta abordagem é uma opção para o desenvolvimento de aplicações científicas.

Geralmente, os cientistas esperam que uma aplicação ou um algoritmo específico aten-

[1]Pontos de variação são utilizados para representar a variabilidade existente entre os produtos de uma LPS (REMMEL et al., 2011).

dam às suas necessidades de pesquisa e experimentação, porém, na maioria dos casos, eles precisam especificar experimentos que consistem em várias aplicações e/ou algoritmos executados de forma encadeada. Estas aplicações/algoritmos devem atuar formando um fluxo, ou seja, um *workflow* científico. *Workflows* científicos são diferentes de *workflows* aplicados a sistemas comerciais, pois possuem características extras que lhes adicionam uma maior complexidade, tais como: (i) *workflows* com um grande número de passos; (ii) volatilidade do *workflow*, já que este poderá ser alterado durante o processo de avaliação das hipóteses científicas, (iii) necessidade de parametrização para muitas tarefas (NARDI, 2009).

Analisando as dificuldades em especificar experimentos científicos e considerando a necessidade de composição de aplicações científicas para a sua implementação, constata-se a necessidade de um suporte semântico mais adequado para a fase de análise de domínio. A hipótese deste trabalho é que o uso de *workflows* científicos em uma Linha de Produtos de Software, com o apoio de modelos de *features* associados a ontologias, pode facilitar o desenvolvimento de experimentos, para a criação de uma Linha de Produtos de Software Científico (LPSC).

A abordagem PL-Science é proposta como solução, cujo propósito é auxiliar os cientistas no processo de escolha e definição de aplicações científicas baseadas em *workflows*, de acordo com as necessidades de pesquisa. Com isso, através dos conceitos de LPS, os cientistas poderão seguir os modelos que especificam a linha de produtos e tomar decisões de acordo com suas necessidades. Ao final, através da abordagem PL-Science, cientistas poderão gerar um produto tal como uma aplicação específica, ou também um *workflow* científico composto por aplicações e/ou algoritmos instanciados, sendo esta a contribuição da abordagem proposta.

1.3 OBJETIVOS

Tendo por base os benefícios da utilização de LPS no contexto de famílias de aplicações com um núcleo comum de características, esta dissertação tem como objetivo principal propor uma abordagem cujo foco é o uso conjunto de ontologias e modelos de *features* em uma LPS, de forma a apoiar o desenvolvimento de *workflows* científicos. Através da conexão entre esses modelos, tem-se também como objetivo demonstrar a aplicação da abordagem para LPSC proposta, chamada PL-Science, bem como a análise dos benefícios

derivados da mesma. O objetivo geral desta abordagem pode ser decomposto nos seguintes objetivos específicos:

i) propor uma arquitetura para a implementação de uma LPS voltada para aplicações científicas;

ii) apresentar uma abordagem onde a semântica é ressaltada, para a especificação das variabilidades de uma LPS, através do uso de ontologias em conjunto com modelos de *features*;

iii) implementar uma LPSC, apresentando exemplos de uso, no domínio da Bioinformática (sequenciamento/alinhamento genético).

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em cinco capítulos.

O Capítulo 2 apresenta os principais conceitos relacionados aos assuntos tratados neste trabalho, como a definição de Linha de Produtos de Software, modelos de *features*, ontologias e software científico.

No Capítulo 3, é apresentada a abordagem PL-Science e seus modelos de domínio, bem como sua arquitetura e metodologia. Ainda neste capítulo são apresentados alguns trabalhos relacionados com a abordagem proposta nesta dissertação.

O Capítulo 4 apresenta a aplicação da abordagem PL-Science a um domínio específico (sequenciamento/alinhamento genético), apresentando exemplos de uso da LPSC criada.

Por fim, o Capítulo 5 contém as contribuições, limitações e trabalhos futuros.

2 PRESSUPOSTOS TEÓRICOS

Este capítulo tem como objetivo apresentar os principais conceitos relacionados ao uso de modelos de *features* e ontologias em Linhas de Produtos Científicas. A Seção 2.1 trata sobre Linha de Produtos de Software (LPS) de forma geral, bem como os aspectos relacionados a seu desenvolvimento. Na segunda seção (2.2) é abordada a questão da variabilidade em LPS e uma forma para representação desta, através de modelos de *features*. Na seção seguinte (2.3), são apresentados os conceitos relacionados a ontologias. Já na Seção 2.4, as principais características de software científico e de *workflows* científicos são apresentadas. Por fim, na Seção 2.5, tem-se as considerações finais sobre este capítulo.

2.1 LINHA DE PRODUTOS DE SOFTWARE (LPS)

Famílias de Software, Famílias de Produtos ou Linhas de Produtos de Software podem ser definidas, segundo CLEMENTS e NORTHROP (2002), como: "[...] um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um segmento particular de mercado ou missão, e que são desenvolvidos a partir de um conjunto comum de ativos principais e de uma forma preestabelecida". Já POHL et al. (2005) definem LPS como um "conjunto de aplicações desenvolvidas utilizando plataformas e customização em massa".

Muitas vezes confunde-se o que realmente é ou não uma LPS, porém, LPS possue características específicas que denotam diferença entre esta abordagem e o desenvolvimento baseado em reúso ou em componentes, tais como (CLEMENTS; NORTHROP, 2001):

- Em uma LPS, a construção de um novo produto ou sistema está mais relacionada à montagem ou à geração do que à criação do produto, já que em uma LPS existe um guia pré-definido que especifica exatamente a forma de construção deste produto;
- LPS propicia economia de escopo, já que os produtos desenvolvidos são similares, porém, deve-se ressaltar que este alto grau de similaridade não ocorre por acaso, ele é resultado de um planejamento que deve ser realizado com esta finalidade;
- Apesar de LPS envolver reúso, este não ocorre por conveniência, ou seja, oportunamente. LPS aborda a reutilização de forma sistemática, em seu mais alto nível,

sendo planejada e executada no desenvolvimento dos produtos ou sistemas;

- Um produto ou sistema que se assemelha a outro e necessita ser tratado separadamente não caracteriza uma LPS e sim reúso de software de forma *ad hoc*. Em LPS maduras, o conceito de múltiplos produtos desaparece, sendo cada produto considerado como pertencente a uma base comum com algumas características diferentes;

- LPS podem envolver o desenvolvimento baseado em componentes, porém é mais do que isto; LPS fazem uso de componentes, porém a utilização destes é realizada de forma prescrita e planejada;

- *Releases* ou versões de um único produto não caracterizam uma LPS.

A abordagem de família de produtos melhora tanto a reusabilidade quanto a capacidade de manutenção dos produtos, pois, sob essa abordagem, o reúso é planejado, ativado e executado. Ao invés de copiar e modificar o que for necessário, através desta abordagem, o núcleo de artefatos usados para a criação dos membros da família de software são geridos em conjunto (YU; SMITH, 2009).

Neste contexto de LPS, variabilidade [1] é a capacidade que um sistema possui de ser eficientemente aumentado, alterado, personalizado ou configurado para o uso em um contexto particular (ASIKAINEN et al., 2007). Produtos que incorporam variabilidades podem apresentar vantagens como abordar vários segmentos e fornecer conjuntos de características diferentes para diferentes necessidades.

Uma *feature* [2] pode ser definida como uma propriedade relevante do sistema, sendo usada para capturar semelhanças e variabilidades entre os produtos de uma LPS (FERNANDES et al., 2008).

2.1.1 DESENVOLVIMENTO DE UMA LPS

O processo de desenvolvimento de uma família de programas é dividido em duas fases: engenharia de domínio e engenharia de aplicação (YU; SMITH, 2009).

A engenharia de domínio também é conhecida como a etapa de desenvolvimento do núcleo de artefatos, enquanto a engenharia de aplicação consiste no desenvolvimento do produto usando o repositório de artefatos da LPS.

[1] Termo herdado do inglês, *variability*.
[2] Os termos *feature* e *features* serão utilizados sem a sua tradução para o Português.

2.1.1.1 Desenvolvimento do Núcleo de Artefatos

Desenvolvimento do Núcleo de Artefatos ou Engenharia de Domínio consiste no desenvolvimento de componentes reutilizáveis, a partir da análise de domínio da LPS.

A utilização de um núcleo de artefatos em uma LPS visa melhorar a produtividade, aumentar a qualidade dos produtos, diminuir os custos de desenvolvimento e o *time to market* para lançar novos produtos ou versões (REINHARTZ-BERGER et al., 2011). Assim, o núcleo de artefatos forma a base de uma LPS e, na maior parte das vezes, inclui a arquitetura e os requisitos, além de componentes reutilizáveis, modelos de domínio, planos de teste e modelos. Estes são utilizados para expressar os pontos comuns e as variabilidades existentes entre os produtos que poderão ser gerados a partir da linha (JUNIOR et al., 2010).

REINHARTZ-BERGER et al. (2011) apresentam uma divisão entre os métodos existentes para a modelagem do núcleo de artefatos de LPS. Os referidos métodos enquandram-se na categoria de Métodos Orientados a *Features* (*Feature-Oriented Methods*), ou na categoria de Métodos de Modelagem Baseado em UML (*UML-based Modeling Methods*), descritas a seguir.

Os Métodos Orientados a *Features* utilizam a composição de *features* em 'árvores', as quais são capazes de expressar as dependências existentes entre as possibilidades de variação dos produtos da LPS. Nesta categoria enquadra-se o método proposto por KANG et al. (1998), que se concentra na especificação dos pontos comuns existentes em uma LPS. Além disso, outras abordagens que se enquadram nesta categoria, permitem a utilização de construções do tipo OR ou XOR, permitindo criar outros relacionamentos entre as *features* previamente estabelecidas. Alguns métodos ((CZARNECKI; KIM, 2005) e (SVAHNBERG et al., 2005)) representam os pontos de variação e as variantes de uma LPS através de grupos de *features*, além de possibilitarem a utilização de construções do tipo OR ou XOR. Nesta categoria, enquadram-se ainda os métodos baseados em cardinalidade (*Cardinality-Based Feature Modeling - CBMF*), na qual também pode ser classificado o método proposto por CZARNECKI e KIM (2005). Tais métodos são capazes de oferecer um pouco mais de expressividade aos métodos puramente orientados a *features*, através da utilização de OCL (*Object Constraint Language*) para especificar dependências entre as *features*, além da definição de cardinalidades diferentes para melhor orientar o desenvolvimento de produtos dentro da LPS. De forma particular, CBMF estende a expressidade

dos diagramas de *features*, que são especificados em FODA (KANG et al., 1990), em cinco aspectos principais. São eles: (i) **cardinalidade**: denota quantos 'clones' de uma determinada *feature* pode ser incluída em um produto; (ii) **grupos de *features***: permite a organização das *features* e a definição de quantos membros de um determinado grupo de *features* podem ser selecionados em pontos do diagrama de *features*; (iii) **tipos de atributos**: servem para indicar que valores dos atributos podem ser especificados durante a configuração do produto; (iv) **referências entre modelos de *features***: possibilita a divisão de um diagrama de *features* em diferentes diagramas; e (v) **relações entre *features***, utilizando em OCL.

Já os Métodos de Modelagem Baseados em UML, em sua maioria, utilizam estereótipos (*stereotypes*) para a diferenciação entre elementos opcionais e obrigatórios da LPS. Alguns métodos desta categoria representam a variabilidade de forma explícita, utilizando, por exemplo, os estereótipos *variation point* e *variant*. Um método que se enquadra nesta categoria é o *Application-based DOmain Modeling* (ADOM) (REINHARTZ-BERGER; STURM, 2009). O ADOM tem como base um perfil (profile) UML que inclui seis estereótipos, sendo eles: *multiplicity*, *variation point*, *variant*, *requires*, *excludes* e *reuse*.

Após a criação do núcleo de artefatos, torna-se possível a sua utilização para a geração dos produtos da LPS. Essa etapa é apresentada a seguir.

2.1.1.2 Desenvolvimento do Produto usando o Núcleo de Artefatos

O processo de Desenvolvimento do Produto, também conhecido como Engenharia de Aplicação, é responsável por analisar os requisitos da aplicação a ser gerada e, logo após, derivar um produto utilizando o modelo de variabilidades. Por fim, o produto gerado é disponibilizado no ambiente do usuário (LINDEN et al., 2007).

A derivação, construção ou desenvolvimento de um produto de software é realizado a partir de uma configuração gerada através do núcleo de artefatos. Esta etapa é realizada pelo engenheiro de aplicação (DEELSTRA et al., 2005). Vale ressaltar que, durante o processo de derivação de um novo produto de software, novos requisitos, que não estavam contemplados na arquitetura da LPS, podem surgir. Assim, para atender a estes novos requisitos, implementações ou adaptações necessitam ser realizadas. Estas adaptações/implementações podem ser feitas diretamente no produto a ser gerado, ou então deve-se realizar uma adaptação do núcleo de artefatos, de forma que este passe a atender

aos novos requisitos.

O desenvolvimento do produto em uma LPS envolve as atividades de análise baseada no modelo do domínio (ou escopo da LPS), instanciação da arquitetura do produto e povoamento da arquitetura com os componentes adequados (ITANA; TRAVASSOS, 2002). Estas atividades são detalhadas a seguir.

A primeira atividade envolve a definição de uma lista de *features* do produto a ser desenvolvido com base no modelo de domínio da LPS. Esta lista engloba tanto as *features* que são recuperadas do modelo de domínio, quanto aquelas que são definidas como novos requisitos a serem integrados para formar a especificação de um novo produto. Estes novos requisitos utilizados devem realimentar o modelo de domínio e o repositório de componentes existentes anteriormente.

A segunda atividade está relacionada à instanciação da arquitetura do produto. A arquitetura genérica possui alguns pontos de variação não resolvidos. Portanto, é necessário um processo iterativo de transformações nesta arquitetura, até que se obtenha a arquitetura específica do produto desejado, de acordo com os requisitos preestabelecidos.

Já a terceira atividade está relacionada à inclusão dos componentes adequados à arquitetura que foi previamente definida. Estes componentes podem ser oriundos do próprio repositório da LPS, pode ser um novo componente que foi implementado para atender aos requisitos necessários ou um componente de prateleira (ou COTS - *commercial off-the-shelf*) que foi adquirido. Além disso, devem ser realizados testes de unidade e verificação de componentes, integração e verificação geral do novo produto.

Uma das formas existentes para facilitar o processo de derivação de produtos de uma LPS é através da utilização de ferramentas de instanciação, que facilitam a seleção, a composição e a configuração dos artefatos do núcleo e de suas respectivas variabilidades. Algumas ferramentas para derivação automática de produtos têm sido propostas, dentre as quais podem ser citadas: pure::variants (BEUCHE, 2011), Gears (KRUEGER, 2010) e Gen-Arch (CIRILO et al., 2007).

2.2 VARIABILIDADE E MODELOS DE *FEATURES*

A identificação explícita de variabilidades entre os produtos de software é um ponto chave que diferencia uma abordagem de desenvolvimento que usa Linha de Produtos de Software de outras práticas de desenvolvimento (LIVENGOOD, 2011).

O gerenciamento de variabilidades também exerce um papel importante em uma LPS, pois abrange as atividades de representar explicitamente as variabilidades dos artefatos que compõem o núcleo da LPS, de gerenciar as dependências entre as diferentes variabilidades e de apoiar as instâncias (ou produtos) geradas a partir da LPS. Variabilidade e gerenciamento de variabilidade são as principais características que distinguem a engenharia de família de produtos de software de outras abordagens para o desenvolvimento de software (ASIKAINEN et al., 2007).

Deve-se ressaltar a existência de duas visões de variabilidade diferentes: a visão do cliente e a visão do desenvolvedor. Os clientes buscam obter aplicações customizadas para as suas necessidades individuais, evidenciando a necessidade de que este conheça pelo menos uma parte das possibilidades de variação existente entre os produtos da LPS. Por outro lado, variabilidade é parte integrante dos artefatos de domínio, portanto, deve ser levada em consideração por quem desenvolve a LPS. Do ponto de vista do desenvolvedor, existe um maior esforço, já que há a necessidade de desenvolvimento de um conjunto de artefatos que possam ser configuráveis, ou seja, considera-se que tais artefatos devem ser desenvolvidos de forma a possibilitarem a geração de diferentes produtos na LPS. Comparando-se com o esforço de desenvolver artefatos específicos para cada aplicação possível, este torna-se consideravelmente menor, já que os artefatos são desenvolvidos especificamente para uma aplicação, sem a necessidade de serem adaptados para inclusão na geração de diversos produtos da LPS. Para diferenciar essas duas visões existe a distinção entre variabilidade interna e externa. A primeira representa a variabilidade que é escondida dos clientes e a segunda, a variabilidade que é visível a estes (POHL et al., 2005).

Modelos de variabilidade são fundamentais para o desenvolvimento e gestão de Linhas de Produtos de Software. Eles podem conter conceitos relacionados a decisões, *features* ou pontos de variação. Modelos de *features* são uma das notações proeminentes usadas para a modelagem de variabilidades (ANDERSEN et al., 2012). Assim, no desenvolvimento de uma LPS, uma das primeiras atividades a serem realizadas é a análise de *features*, que identifica as características externamente visíveis dos produtos da LPS e as organizam em um modelo de *features* (LEE; MUTHIG, 2006). É através da modelagem de *features* que são especificadas as funcionalidades comuns e variáveis da família de aplicações a serem geradas por meio da linha de produtos.

Como resultado da modelagem de *features*, obtém-se um modelo que inclui todos

os pontos de variação (onde as características entre os produtos da linha podem sofrer variações) e as variantes (possíveis valores de um ponto de variação) dos artefatos de uma LPS. Este modelo também deve incluir restrições entre os pontos de variação e variantes, já que um ponto de variação (ou uma variante) pode exigir ou excluir outro ponto de variação (ou uma variante) (REMMEL et al., 2011).

A abordagem FODA (*Feature-Oriented Domain Analysis*) (KANG et al., 1990) foi uma das primeiras técnicas para modelagem de *features* relatados na literatura, tendo as notações de modelos mais recentes muitos pontos em comum com essa abordagem.

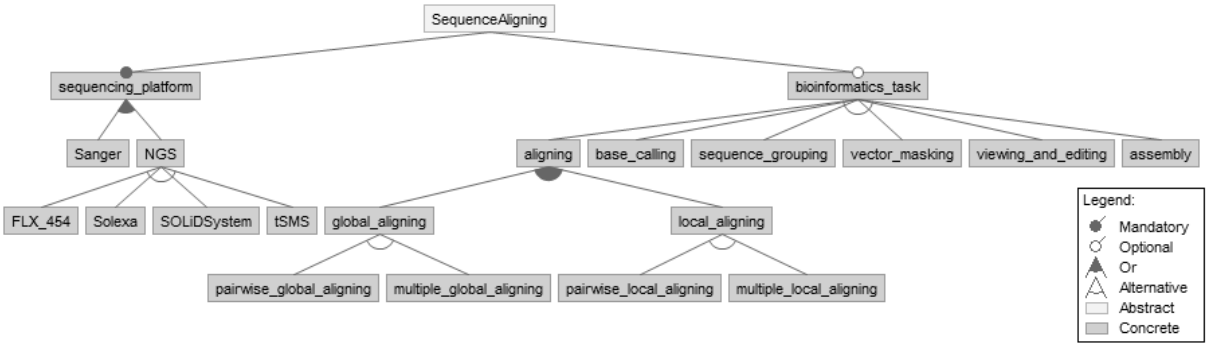

Figura 2.1: Exemplo de Modelo de *Features*

Um modelo de *features* em FODA é representado por um grafo acíclico, direcionado, com uma raiz conforme exibido na Figura 2.1. A raiz representa um conceito e possui *subfeatures*. Uma *subfeature*, por sua vez, pode ter outras *subfeatures*. As *subfeatures* dividem-se em: (i) obrigatórias (*mandatory*): devem ser selecionadas sempre que a *subfeature* pai é selecionada; (ii) opcionais (*optional*): podem ser selecionados sempre que a *feature* pai é selecionada, mas não obrigatoriamente; e (iii) alternativas (*alternative*): consiste em um conjunto de *subfeatures* do qual exatamente uma deve ser selecionada sempre que a *feature* pai é escolhida (KANG et al., 1990). O modelo de *features* também possibilita a criação de *features* alternativas exclusivas (*or*): se uma das *subfeatures* é selecionada, as *features* que se encontram no mesmo nível desta e que possuem o mesmo pai são excluídas. Este tipo de *feature* é usado apenas quando a mesma possui apenas duas *subfeatures* diretamente associadas.

Na Figura 2.1 [3], é apresentado parte de um modelo de *features* no domínio da Bioinformática, que trata as variabilidades existentes no contexto de sequenciamento/alinhamento genético. De acordo com esta figura, a *feature* raiz *SequenceAligning* representa

[3]Cabe ressaltar que esta figura foi extraída de parte do modelo de *features* utilizado para a implementação dos cenários de uso da abordagem PL-Science.

um conceito, ou seja, Alinhamento de Sequências. As *subfeatures* definidas abaixo dela representam as possibilidades de variação existentes neste domínio. Conforme pode ser visualizado, a *feature sequencing_platform* é obrigatória (*mandatory*). Isto significa que, para o domínio de Alinhamento de Sequências, deve-se definir uma plataforma de sequenciamento a ser utilizada. Como *subfeatures* de *sequencing_platform*, tem-se as *features Sanger* e *NGS*. Por serem *features* alternativas exclusivas (*or*), a seleção de uma implica a exclusão da outra. O mesmo ocorre com as *subfeatures* alternativas (*alternative*) da *feature NGS*. Como exemplo de *feature* opcional (*optional*), tem-se a *feature bioinformatics_task*.

Além dos tipos de *features* apresentados anteriormente, utilizando a notação FODA (KANG et al., 1990), é possível modelar regras de dependência entre *features*, que podem ser de dois tipos:

- *Requires* (Requer): Este tipo de regra especifica que a escolha de uma *feature* implica na escolha de outra. Por exemplo, dadas duas *features* A e B: se A Requires B, escolher a *feature* A implica automaticamente escolher a *feature* B.

- *Excludes* (Exclui): Este tipo de regra especifica que a escolha de uma *feature* implica na exclusão de outra. Por exemplo, dadas duas *features* A e B, se A Excludes B, a escolha da *feature* A exclui a possibilidade de escolha da *feature* B

Além da abordagem FODA (KANG et al., 1990), diversas abordagens baseadas em *features* têm sido propostas ao longo dos anos, tais como: FORM (*Feature-Oriented ReuseMethod*) (KANG et al., 1998) , PuLSE (BAYER et al., 1999), KobrA (ATKINSON et al., 2000), FOOM (*Feature-based Object Oriented Modeling*) (AJILA; TIERNEY, 2002), Software Engineering Institute's *framework* for SPL (CLEMENTS; NORTHROP, 2001), SPL Framework (POHL et al., 2005), PLUS (GOMAA, 2005), RiDE (ALMEIDA, 2009) e a notação para representação de variabilidade em modelos de *features* Odyssey-FEX, do ambiente Odyssey (ODYSSEY, 2013).

No entanto, apesar da existência das abordagens anteriormente citadas, na tentativa de disponibilizar uma LPS para o domínio de *workflows* científicos, nos deparamos com o problema de qual o tipo de modelo de domínio utilizar. Uma das possibilidades seria justamente a utilização de modelos de *features*. Porém, neste trabalho, a utilização somente de modelos de *features* para a representação das restrições existentes no domínio apresentou-se limitada. Segundo FILHO et al. (2012), estes modelos não foram projetados de forma

que facilitassem a interoperabilidade, a recuperação e a inferência de informações. Tal fato também é ressaltado por (JOHANSEN et al., 2010) e (CZARNECKI et al., 2006). Ou seja, o uso de modelos de *features* apenas não parece suficiente para expressarmos todas as restrições existentes em um determinado domínio (como, por exemplo, no domínio de sequenciamento/alinhamento genético). Eles não oferecem a possibilidade de expressar toda a semântica envolvida nos relacionamentos entre *features* necessários em aplicações científicas.

Para exemplificar, temos que no domínio de sequenciamento/alinhamento genético, conforme o modelo de *features* apresentado na Figura 2.1, não foi possível expressar toda a semântica necessária em aplicações científicas. Estas restrições estão relacionadas sobretudo à semântica envolvida nos relacionamentos entre as *features*. Um maior poder semântico pode ser fornecido a LPS através do uso de ontologias e da possibilidade de inferência oferecida por elas.

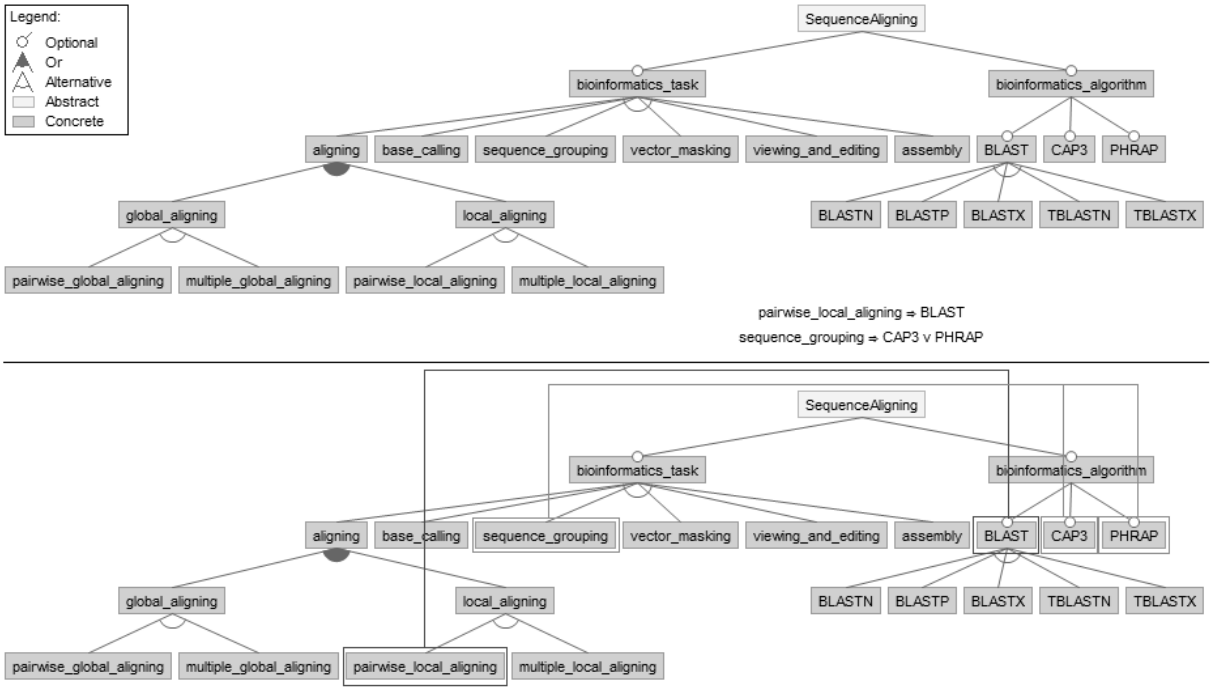

Figura 2.2: Representação de Restrições no Modelo de *Features*

Conforme exibido na Figura 2.2, na parte superior, utilizando apenas modelos de *features*, consegue-se estabelecer algumas relações existentes entre as *features*. Pode-se dizer que a seleção da *feature pairwise_local_aligning* implica na seleção da *feature BLAST*, ou ainda que a seleção da *feature sequence_grouping* implica na seleção de uma das *features CAP3* ou *PHRAP*. Porém, utilizando apenas as restrições possíveis em modelos de *fea-*

tures, não se consegue expressar quando a seleção de uma determinada *feature* torna-se mais adequada do que outra. Por exemplo: como expressar quando a seleção da *feature CAP3* é mais adequada do que a *feature PHRAP* (no caso da *feature sequence_grouping* ter sido selecionada)?

Segundo JOHANSEN et al. (2010), um dos fatores que implicam na escolha de modelos de *features* para a representação de variabilidades refere-se ao fato de que tanto os conceitos como as relações expressas neste tipo de modelo não são totalmente definidas, ou, às vezes, são definidas de forma insatisfatória. Este trabalho ressalta ainda que modelos de *features* são, na maior parte das vezes, demasiadamente vagos para serem analisados por uma máquina de inferência.

Apesar das deficiências apresentadas com relação a modelos de *features*, vale ressaltar que existe a possibilidade de agregar mais semântica aos modelos de *features*, adicionando a estes assertivas descritas em lógica proposicional. A ferramenta Gears (KRUEGER, 2010), por exemplo, sugere que estas regras de associação e de dependência entre as *features* possam ser expressas através deste mecanismo de assertivas baseadas em lógica descritiva. Embora linguagens baseadas em lógica proposicional também ofereçam uma alta expressividade, a sua utilização por pessoas que não possuem muita familiaridade com a notação da linguagem escolhida não é trivial.

Outra forma que pode ser utilizada para representação formal das restrições entre os pontos de variação de uma LPS é através de ontologias. Ontologias permitem a utilização de mecanismos de inferências, através dos quais se consegue obter novos conhecimentos, sendo esta uma das grandes vantagens de sua utilização. Além disso, a criação de restrições ao utilizarem-se ontologias (descritas usando OWL-DL) é mais simples do que a criação de assertivas em lógica proposicional.

2.3 ONTOLOGIAS

Em Ciência da Computação, uma ontologia define uma especificação formal e explícita de uma conceitualização compartilhada (GUARINO, 1998). Ela permite capturar o entendimento comum de objetos e seus relacionamentos em um determinado domínio. Ontologias fornecem um modelo formal e manipulável deste domínio (GRUBER, 1995). Sua utilização reúne benefícios como reúso, compartilhamento de conhecimento, portabilidade, manutenção e confiabilidade, partindo-se do princípio que elas representam uma concei-

tuação compartilhada.

Com o uso de ontologias, consegue-se estabelecer uma compreensão comum sobre objetos e os relacionamentos existentes entre eles em um determinado domínio, através de um modelo formal e manipulável. Além disso, a especificação formal do significado dos termos envolvidos na ontologia possibilita a criação de novos termos, através da combinação dos já existentes (MATOS, 2008). Ontologias são uma das abordagens que podem ser utilizadas para a representação do conhecimento. Além disso, elas permitem que o conhecimento sobre um domínio possa ser lido e processado por um sistema computacional. Adicionalmente, há a possibilidade de utilização de máquinas de inferências (*reasoners*), que oferecem algoritmos através dos quais se consegue derivar novas informações e relacionamentos que anteriormente estavam implícitos na ontologia incial (FILHO et al., 2012).

O uso de ontologias na área de software popularizou-se com a Web Semântica. Assim, ontologias passaram a ser usadas como uma taxonomia e um conjunto de regras de inferência, através das quais se torna possível capturar o conhecimento que não está explícito na taxonomia (BERNERS-LEE et al., 2001). De acordo com o W3C (*World Wide Web Consortium*), a linguagem padrão para definição de ontologias é a OWL (*Web Ontology Language*) (BECHHOFER et al., 2004). Esta linguagem foi projetada para ser utilizada por aplicações que necessitam processar o conteúdo da informação ao invés de somente apresentar informações para os humanos. Esta linguagem tem como objetivo suprir as necessidades de uma linguagem de ontologia para a Web, além de resolver algumas limitações das linguagens anteriores, como XML e RDF.

Utilizando OWL consegue-se descrever classes, descrições de classes, propriedades e suas instâncias. Além disso, como esta linguagem é baseada em lógica descritiva, é possível a utilização de mecanismos de inferência, os quais permitem explicitar conhecimentos que estão implícitos em uma base de conhecimento. Assim, ontologias descritas utilizando OWL não devem ser consideradas apenas sob o ponto de vista de sua sintaxe, mas também de sua semântica.

Enquanto modelos de *features* são mais intuitivos e mais simples de construir, o uso de ontologias para a representação de LPS proporciona um maior formalismo, além do alto poder de descrição semântica de ontologias descritas utilizando OWL. Também podem ser citadas como vantagens do uso de ontologias as possibilidades de recuperação de

informações, de inferência e de rastreabilidade (FILHO et al., 2012).

2.3.1 INFERÊNCIA

Uma das características mais relevantes em uma ontologia é a possibilidade de inferir novos conhecimentos através do uso de máquinas de inferência (*reasoners*). Neste contexto, uma máquina de inferência consegue inferir, por exemplo, uma hierarquia de acordo com o que foi definido na ontologia. Assim, a máquina de inferência pode ser utilizada para testar se uma determinada classe é uma subclasse de outra classe declarada na ontologia. Outra funcionalidade que é oferecida pelas máquinas de inferência é a verificação de consistência da ontologia. Com base na descrição (restrições) de uma classe, a máquina de inferência pode verificar se é ou não possível para uma classe possuir quaisquer instâncias. Uma classe é considerada inconsistente quando a mesma não pode ter nenhuma instância. Este tipo de verificação é possível graças ao processamento das restrições relacionadas às classes que foram declaradas na ontologia (HORRIDGE; BRANDT, 2011).

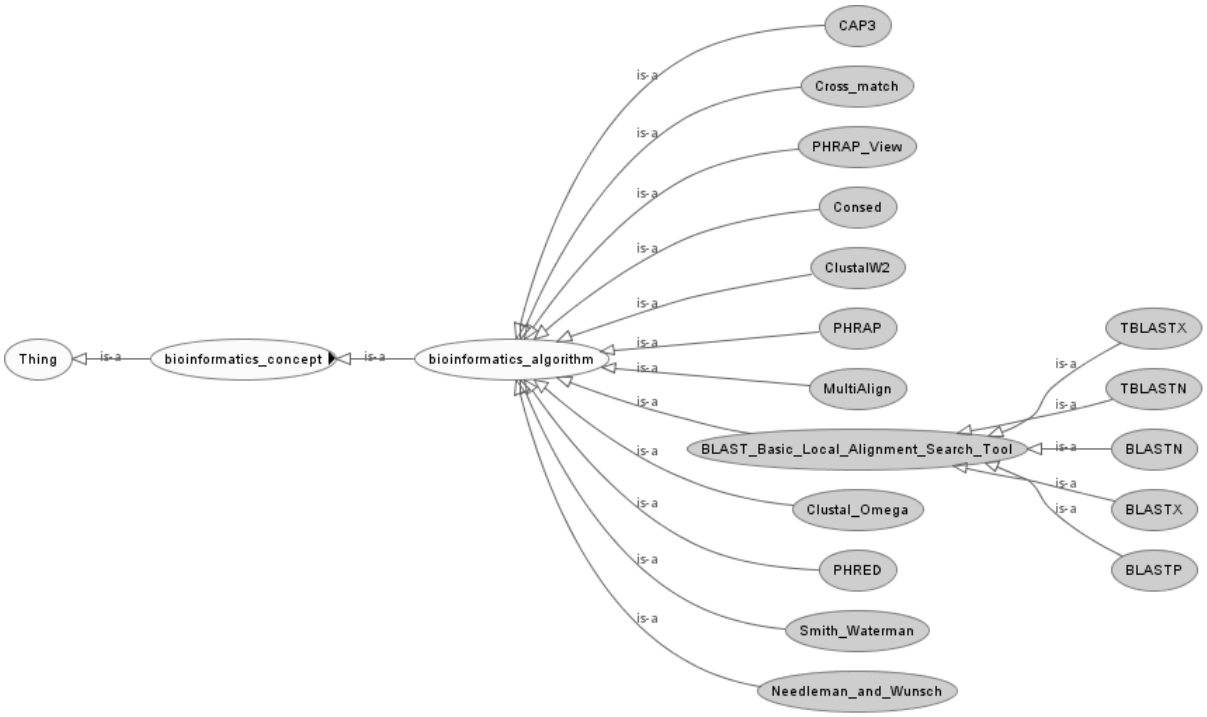

Figura 2.3: Modelo Definido (ou Declarado)

Como exemplo de utilização do mecanismo de inferências tem-se as Figuras 2.3 e 2.4, as quais apresentam uma pequena parte da ontologia de Alinhamento de Sequências (*Sequence Alignment Ontology*), utilizada para a implementação da abordagem PL-Science.

Na Figura 2.3, tem-se a hierarquia de classes conforme as mesmas foram declaradas (ou seja, o modelo definido). Nota-se nesta figura que as classes *ClustalW2* e *Clustal_Omega* são subclasses da classe *bioinformatics_algoritms*. Esta mesma situação ocorre com as classes *PHRAP_View* e *Consed*, e também com as classes *CAP3* e *PHRAP*. Após a utilização de uma máquina de inferência[4] sobre a mesma ontologia, que continha as classes anteriormente citadas, consegue-se visualizar o que é exibido na Figura 2.4, ou seja, de acordo com as restrições existentes entre as classes, consegue-se visualizar que a classe *ClustalW2* passa a ser uma subclasse de *Clustal_Omega* (Figura 2.4(a)). Já na Figura 2.4(b), tem-se que as classes *PHRAP_View* e *Consed* são semelhantes. O mesmo ocorre com as classes exibidas na Figura 2.4(c).

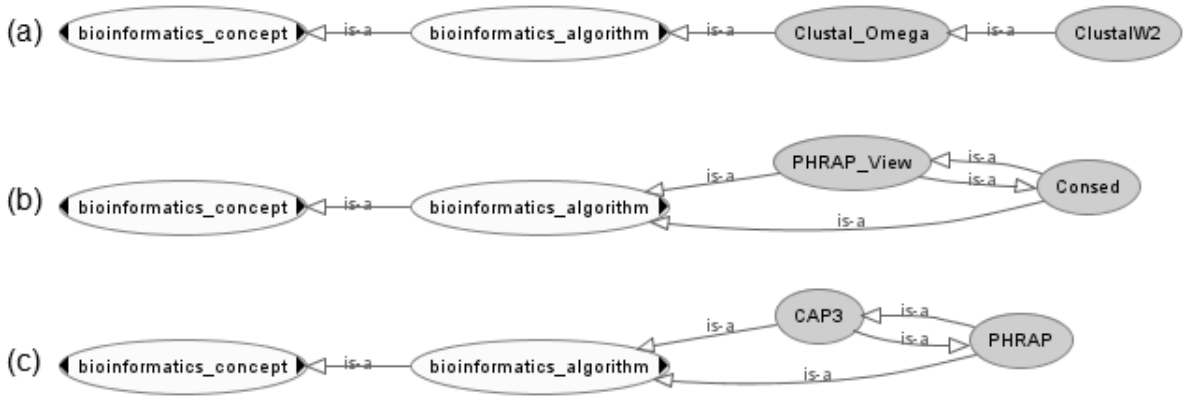

Figura 2.4: Modelo Inferido

Como um exemplo das vantagens do uso de mecanismos de inferência (Figura 2.4(a)), no caso do domínio em questão, podemos citar a facilidade de obter a informação de que a classe *ClustalW2*, que representa uma das *features* relacionadas aos algoritmos para sequenciamento/alinhamento genético, é uma subclasse ou uma especificação da classe *Clustal_Omega*. Isto significa que as instâncias relacionadas à classe *ClustalW2*, na ontologia, possuem as características das instâncias relacionadas à classe *Clustal_Omega*. Ou seja, dependendo da configuração do produto a ser gerado dentro da LPSC proposta, pode-se avaliar a opção da utilização de um algoritmo que possui a *feature Clustal_Omega* no lugar de um algoritmo que possui a *feature Clustal_W2*.

Levando em consideração que ontologias podem ser usadas para modelar um domínio específico, elas podem também melhorar a modelagem de *features*, fornecendo informações adicionais para o domínio da LPS, principalmente com relação à especificação de restrições

[4]Foi utilizado o *reasoner* FaCT++ (FACT++, 2013).

entre as *features* existentes. Ou seja, em ontologias descritas em OWL, as *features* podem ser expressas como classes desta ontologia e a forma como estas *features* se relacionam, bem como suas restrições de utilização, são expressas através das propriedades ligadas às classes (*object properties*).

2.4 SOFTWARE CIENTÍFICO

A pesquisa científica obteve grandes avanços através do apoio proporcionado pela computação. Além dos pilares da teoria e da experimentação, a computação passou a ser reconhecida como o "terceiro pilar" a apoiar a ciência (CARVALHO et al., 2006). De acordo com a Figura 2.5, o primeiro pilar representa uma determinada teoria ou um modelo específico em um campo de pesquisa, enquanto o segundo pilar representa os experimentos a serem realizados. O terceiro pilar é representado por técnicas que permitem simulações computacionais baseadas na eficiência de métodos numéricos e leis físicas conhecidas (RIEDEL et al., 2008).

Figura 2.5: Computação e o Terceiro Pilar [adaptado de (RIEDEL et al., 2008)].

Pesquisas em Engenharia de Software, tradicionalmente, focam em técnicas, métodos e conceitos que podem ser aplicáveis em um contexto geral. Porém, software científicos possuem domínios muito especializados, e, portanto, nem todas as técnicas aplicáveis a software em geral trazem bons resultados em aplicações científicas (SLETHOLT et al., 2011), o que fomenta a necessidade de pesquisas mais específicas nesta área, ou seja, pesquisas que se concentrem na aplicação de técnicas relacionadas à Engenharia de Software no contexto das pesquisas científicas.

Software científicos são usados para fazer cálculos complexos, simulações ou para testar uma teoria científica. Neste contexto, muitas vezes não se pode determinar qual a saída

correta de um aplicativo. Além disso, o software pode evoluir ao longo dos anos, com novas funcionalidades acrescentadas ao sistema. Devido à natureza exploratória dos ambientes científicos, o levantamento e especificação de requisitos são problemáticos porque eles podem ser pouco claros, ou mesmo desconhecidos (SLETHOLT et al., 2011).

Software científicos são fundamentalmente diferentes de software tradicionais, devido, principalmente, aos seguintes aspectos: (i) existe informalidade no processo de desenvolvimento de software científico; (ii) geralmente, os próprios pesquisadores e/ou cientistas desenvolvem o software; (iii) o levantamento e a especificação de requisitos são problemáticos, pois estes podem não se apresentar de forma clara, ou às vezes são até desconhecidos, em um momento inicial da pesquisa (SOUZA, 2011).

Considerando o domínio científico de sequenciamento/alinhamento genético (PROSDOCIMI et al., 2002), uma série de tarefas deve ser realizada de forma a obter-se uma correspondência genética entre duas ou mais sequências. A verificação de tal correspondência entre sequências consiste em identificar regiões similares de sequências de DNA, RNA ou proteínas que possam ser consequência de relações funcionais, estruturais ou evolucionárias entre elas. As tarefas necessárias para obtenção e comparação destas sequências podem ser organizadas em um fluxo de tarefas científicas, ou seja, em um *workflow* científico.

A importância do uso de *workflows* científicos está inerentemente relacionada à forma como atualmente os cientistas elaboram um experimento científico *in silico*[5] e à necessidade atual de colaboração entre cientistas de diferentes centros de pesquisa.

Por conta dessa necessidade de colaboração, é imperativo se organizar um fluxo de execução de aplicações científicas, que devem ser sequenciadas de forma a realizar o experimento. Desta forma, entre uma das atividades que devem ser desenvolvidas por cientistas/pesquisadores encontra-se o sequenciamento (ou composição) de programas/aplicações científicas, onde cada um destes programas produz uma coleção de dados com uma determinada semântica e sintaxe. Esta coleção de dados, na maior parte das vezes, pode ser utilizada como entrada de dados para o próximo programa. Porém, deve-se ressaltar que esta composição de programas não é uma tarefa trivial, o que acaba tornando-se uma barreira para análises mais detalhadas por parte dos pesquisadores. Uma das formas para minimizar tal problema é através da utilização de *workflows* científicos, ou seja, os

[5]Expressão utilizada para denotar simulações computacionais que modelam um processo natural ou de laboratório (LEMOS, 2004).

experimentos *in silico* são representados por meio do encadeamento de atividades, sendo que cada uma das atividades é mapeada para uma aplicação, formando um fluxo coerente de informações e controles. Nele, os dados de saída de um programa são entradas do próximo programa. Este encadeamento de atividades é chamado de *workflow* científico (SILVA, 2011b).

Workflows fornecem meios sistemáticos e automatizados para a condução de análises de conjuntos de dados e integração de aplicações. Através da utilização destes, torna-se possível capturar processos cujos resultados poderão ser reproduzidos e os métodos poderão ser revisados, validados, repetidos e/ou adaptados. Assim, um *workflow* oferece uma interface por meio da qual os pesquisadores/cientistas podem criar fluxos de trabalho a serem realizados, porém, sem a necessidade de programação de baixo nível (GOBLE; ROURE, 2009).

Workflows científicos diferem-se daqueles aplicados a sistemas de negócios por possuírem características que lhes acrescentam maior complexidade, tais como: (i) fluxos com um grande número de etapas; (ii) volatilidade dos fluxos, que devem poder ser alterados frequentemente, durante o processo de avaliação de hipóteses científicas; (iii) necessidade de parametrização para grande número de tarefas (NARDI, 2009). Considerando a volatilidade dos fluxos científicos, tem-se como proposta abordá-los como uma LPS, permitindo assim que diferentes fluxos em um determinado domínio possam ser criados, de acordo com as necessidades do cientista e com o desenvolvimento do experimento científico e o surgimento de novos requisitos relacionados ao experimento.

Como *workflows* científicos estão estreitamente relacionados com os experimentos científicos propriamente ditos, os *workflows* possuem como vantagem a possibilidade de serem executados inúmeras vezes, com pequenas modificações nos programas ou nos parâmetros operacionais do *workflow*. Este processo, independentemente da natureza do *workflow*, é interativo e incremental (MATTOS et al., 2008).

Workflows científicos podem ser considerados cruciais na interação de cientistas com a infraestrutura computacional aplicada à pesquisa, pois facilitam as atividades de *e-Science* e permitem que os cientistas modelem, executem, verifiquem, reconfigurem e reexecutem as análises de dados científicos (SILVA, 2010).

O processo de definição de um *workflow* científico é dependente do domínio de aplicação do *workflow* em questão. Cada *workflow* requer uma ampla discussão entre os membros

do grupo de pesquisa relacionado ao domínio envolvido. Sua definição geralmente requer a tomada de decisão e análises refinadas sobre cada uma de suas etapas (MATTOS et al., 2008).

2.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou os principais conceitos relacionados ao desenvolvimento da abordagem PL-Science, tais como LPS, variabilidades, modelos de *features* e ontologias. Também foram detalhadas as principais características de software científico, principalmente de *workflows* científicos.

3 A ABORDAGEM PL-SCIENCE

Neste capítulo será descrita a abordagem PL-Science, sendo detalhada sua arquitetura, modelos utilizados e também a metodologia de utilização da mesma. Outras abordagens relacionadas à proposta deste trabalho são apresentadas e comparadas com a abordagem PL-Science na Seção 3.5. Ao final, são apresentadas as considerações finais deste capítulo.

3.1 INTRODUÇÃO

Usuários de aplicações científicas, na maioria das vezes, trabalham em um campo específico de investigação, e, nem sempre, têm um treinamento adequado para desenvolvimento de software. Muitas vezes eles iniciam uma nova aplicação copiando um aplicativo existente e realizam pequenas alterações neste de forma a atender às suas necessidades. Além disso, quando o objetivo é especificar um experimento científico, muitas vezes várias aplicações científicas precisam ser combinadas e compostas, formando um *workflow* científico, que em geral é executado de maneira distribuída. Neste contexto de uso de *workflows* científicos, a especificação e implementação das funcionalidades necessárias se torna ainda mais complexa, o que muitas vezes inibe o cientista quanto a utilização de aplicações científicas na especificação e execução de seu experimento.

Este capítulo apresenta a abordagem PL-Science, a qual considera o contexto de Linha de Produtos de Software (LPS). A abordagem tem como objetivo auxiliar os cientistas na definição de um experimento científico, através da especificação de um *workflow*, que englobe as aplicações científicas que devem ser encadeadas para a realização do dado experimento. Usando os conceitos propostos em LPS, os cientistas poderão utilizar os modelos que especificam a linha de produtos científica e, cuidadosamente, tomar decisões de acordo com suas necessidades.

No contexto deste trabalho, Linhas de Produtos de Software Científicas diferem-se de Linhas de Produtos de Software em geral devido ao fato de que as Linhas de Produto de Software Científicas (LPSC) fazem uso de um modelo abstrato de *workflow* científico. Este *workflow* é definido de acordo com o domínio científico em questão e, a partir deste modelo abstrato de *workflow*, serão instanciados os produtos desta linha (Figura 3.1). No entanto, é importante ressaltar que nada impede que o cientista, ao utilizar uma LPSC, especifique

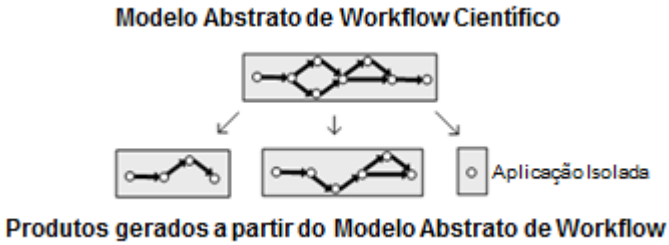

Figura 3.1: Modelo Abstrato de *Workflow* Científico

apenas uma ou mais aplicações científicas a serem utilizadas de maneira isolada, ou seja, sem a utilização de um *workflow* científico, uma vez que a LPSC provê suporte adequado para esta tarefa.

Assim, considerando os benefícios da utilização de LPS no contexto de aplicações que compartilham um núcleo de artefatos comum, propõe-se o uso de uma LPSC, voltada ao apoio de experimentos científicos no domínio de sequenciamento genético (PROSDOCIMI et al., 2002). No contexto da proposta desta LPSC para o domínio de sequenciamento genético, esta dissertação enfoca a etapa de engenharia de requisitos em LPSC, mais especificamente considerando a análise de domínio de uma LPS. Analisando as dificuldades de especificação de experimentos científicos, considerando a composição de aplicações científicas para sua execução, um apoio semântico na etapa de análise de domínio se faz necessário. De acordo com o disposto, este trabalho propõe a conexão de modelos de *features* e ontologias, de forma a combinar os seus benefícios para a modelagem de domínio (COSTA et al., 2012b). A hipótese é que o uso de *workflows* científicos em uma Linha de Produtos de Software, com o apoio de modelos de *features* associados a ontologias pode facilitar o desenvolvimento de experimentos científicos, para a criação de uma Linha de Produtos de Software Científico (LPSC). Desta forma, tem-se como objetivo final da LPSC a geração de um *workflow* científico com as atividades a serem realizadas de acordo com o domínio escolhido e com as necessidades do usuário, ou então a sugestão de aplicações científicas adequadas a uma determinada tarefa científica.

3.2 MODELOS DE DOMÍNIO NO CONTEXTO DA PL-SCIENCE

No contexto de uma LPSC, o entendimento do domínio e a seleção das características desejáveis em um *workflow* e/ou aplicação científica é de grande importância, uma vez que uma das dificuldades do cientista é o desenvolvimento deste tipo de aplicação. A

seguir é apresentado o detalhamento do modelo de *features* e da ontologia no contexto do subdomínio de sequenciamento genético.

3.2.1 MODELO DE *FEATURES*

O modelo de *features* de uma LPS tem por objetivo apresentar as características comuns e variáveis de um domínio. Neste trabalho, o modelo de *features* será subdividido em diversas figuras, a fim de facilitar a apresentação das variabilidades existentes no domínio de sequenciamento/alinhamento genético.

O sequenciamento de DNA consiste em uma série de métodos bioquímicos cuja finalidade é determinar a ordem das bases nitrogenadas (adenina - A, guanina - G, citosina - C e timina - T) da molécula de DNA (DARNELL J. E., 1986). Existem vários métodos ou plataformas disponíveis para a realização do sequenciamento de DNA, sendo que, cada uma delas apresenta suas vantagens e desvantagens. Estas plataformas e métodos configuram-se como características relevantes neste domínio e, portanto, devem também ser explicitadas no modelo de *features*, conforme apresentado na Figura 3.2.

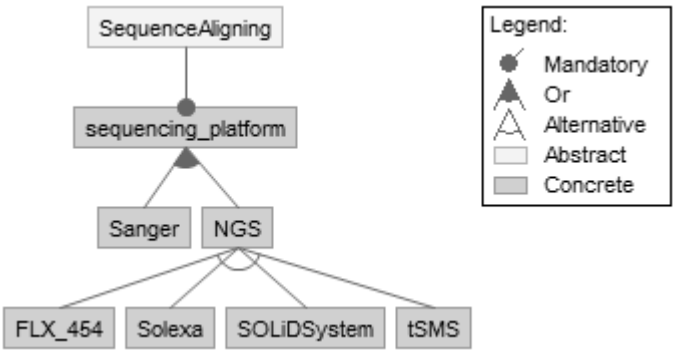

Figura 3.2: Plataformas de Sequenciamento

No método de Sanger (SANGER et al., 1977), o DNA é usado como um *template* para gerar um conjunto de fragmentos que diferem em comprimento uns dos outros, por uma única base nitrogenada. Estes fragmentos são então separados por tamanho e as bases da extremidade são identificadas de forma a recriar a sequência original do DNA. O sequenciamento pelo método de Sanger, além de ser amplamente utilizado por centros de pesquisa, permitiu a execução de diversos projetos-genoma existentes.

Além do sequenciamento pelo método de Sanger, podem ser citadas as novas tecnologias de sequenciamento, chamadas de tecnologias de sequenciamento de nova geração, ou

Next Generation Sequencing (NGS), que começaram a ser comercializadas no ano de 2005. Dentre estas pode-se mencionar a 454 FLX da Roche, a Solexa da Illumina, a plataforma da Applied Biosystems, denominada SOLiD System, e o HeliscopeTrue Single Molecule Sequencing (tSMS), da Helicos. Ambas as plataformas, anteriormente citadas, apresentam um alto poder de gerar informação, quando comparado ao sequenciamento pelo método de Sanger e uma economia de tempo e custo por base de DNA a ser sequenciada. Essa eficiência é proveniente do uso da clonagem *in vitro* e de sistemas de suporte sólido para as unidades de sequenciamento. Com isso, elimina-se o trabalho laboratorial intensivo para produção de clones bacterianos, da montagem das placas de sequenciamento e da separação dos fragmentos em géis (CARVALHO; SILVA, 2010).

Embora sejam plataformas de sequenciamento distintas, tanto o método de Sanger como as novas tecnologias de sequenciamento (NGS) apresentam vantagens e limitações, tais como a relação custo/benefício para obter e gerar os dados necessários. A escolha do método dependerá, principalmente, do objetivo do trabalho a ser desenvolvido pelo cientista. Por exemplo, o método de Sanger apresenta como principais vantagens a geração de *reads* maiores do que em NGS, além da alta precisão da base gerada, no processo de *base calling* (este processo é descrito a seguir). Já as tecnologias de nova geração possuem como principal vantagem a construção *in vitro* de bibliotecas genômicas sem amplificação de fragmentos de DNA e sem clonagem. Por outro lado, estas técnicas de sequenciamento necessitam de equipamentos e kits de reações que apresentam um custo bastante elevado, quando comparado ao método de Sanger.

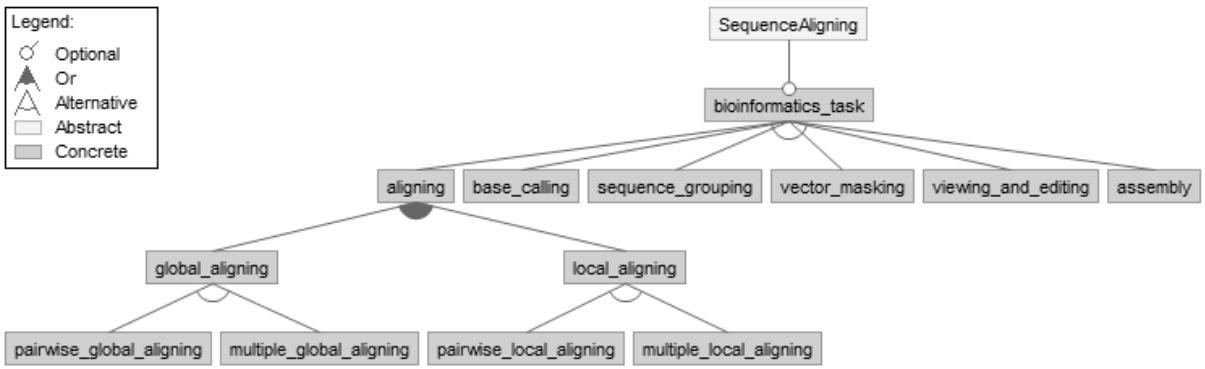

Figura 3.3: Tarefas para Sequenciamento/Alinhamento Genético

Outro conjunto de *features* de grande importância no domínio de sequenciamento/alinhamento genético está relacionado às tarefas que precisam ser realizadas para a obtenção das sequências e também para a comparação das mesmas. Estas tarefas (alinhamento,

base calling, agrupamento de sequências, mascaramento de vetores, visualização e edição e montagem) foram incluídas no modelo de *features* e são apresentadas na Figura 3.3. Estas *features* são brevemente apresentadas a seguir e um maior detalhamento destas é apresentado na Seção 4.2, que explora o domínio de sequenciamento/alinhamento genético.

Resumidamente, a tarefa de *base calling* engloba o processo de leitura dos dados gerados por um sequenciador e o processo de identificação da sequência de DNA gerada, atribuindo um valor de qualidade para cada posição nucleotídica identificada (PROSDOCIMI et al., 2002).

A tarefa de mascaramento de vetores (*vector_masking*) consiste em realizar uma busca de regiões na sequência produzida, as quais devem ser retiradas ou mascaradas, já que estas regiões não representam o DNA que se deseja analisar (PROSDOCIMI et al., 2002).

O agrupamento de sequências (*sequence_grouping*) também é conhecido como *clusterização*. Durante esta etapa, os fragmentos do DNA advindos do sequenciamento são agrupados, formando um *contig* (pequenos fragmentos de DNA sequenciados em sequências maiores) (PROSDOCIMI et al., 2002).

Já a fase de visualização e edição (*viewing_and_editing*) trata-se de uma inspeção manual para verificação da montagem ou agrupamento das sequências.

O alinhamento (*aligning*) consiste, basicamente, em colocar uma sequência sobre a outra e compará-las, com o objetivo de especificar as partes comuns às sequências. O alinhamento entre duas (alinhamento duplo) ou mais sequências (alinhamento múltiplo) pode ainda ser dividido em local e global, conforme exibido na Figura 3.3.

Outra *feature* apresentada como *subfeature* de *bioinformatics_task* é a montagem (*assembly*) das sequências após o alinhamento.

3.2.2 ONTOLOGIA

A ontologia utilizada neste trabalho é uma adaptação da ontologia *myGrid* (WOLSTENCROFT et al., 2007), especificada usando OWL (BECHHOFER et al., 2004). A ontologia *myGrid* foi escolhida como base pois descreve o domínio da Bioinformática e as dimensões com as quais um serviço pode ser caracterizado a partir da perspectiva do cientista. Para a elaboração da ontologia foram reutilizadas partes da ontologia *myGrid* relacionadas ao domínio de sequenciamento/alinhamento genético e, além disso, foram adicionadas

algumas extensões, a fim de possibilitar o uso desta ontologia em conjunto com o modelo de *features* da LPSC. A principal modificação realizada na ontologia foi a adição de restrições entre termos ontológicos importantes no domínio de sequenciamento genético (exemplos destas restrições são apresentados na Figura 3.4), de forma que a semântica destas restrições pudesse ser expressa na ontologia *myGrid*.

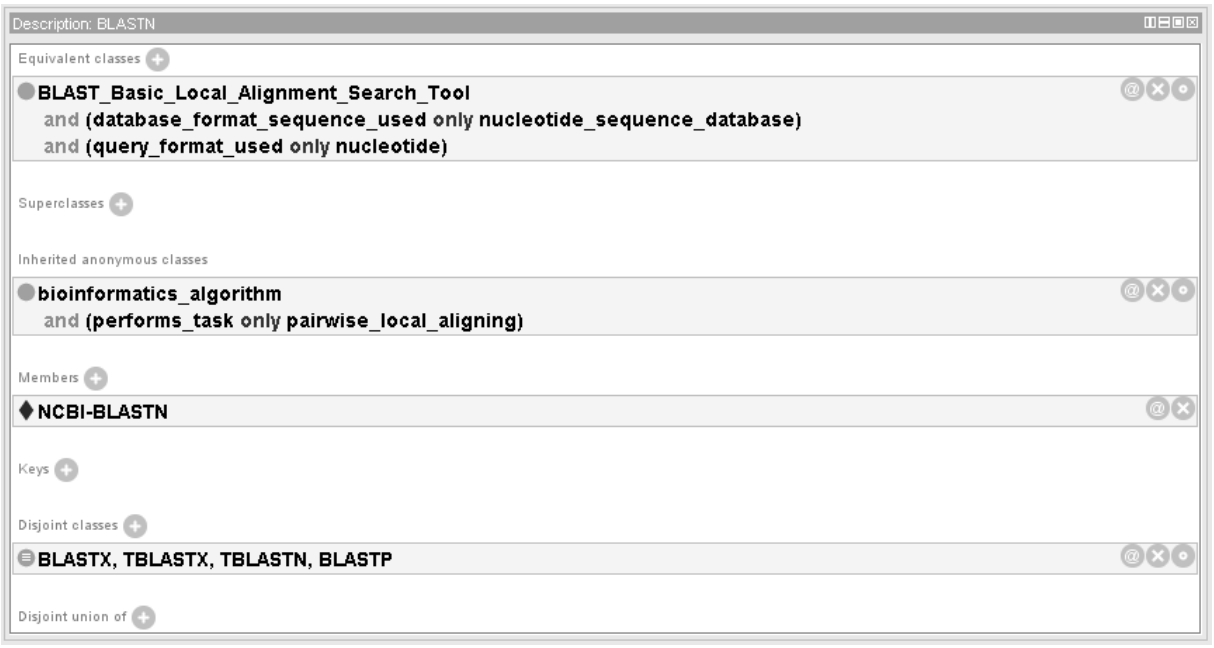

Figura 3.4: Restrições da Classe *BLASTN*

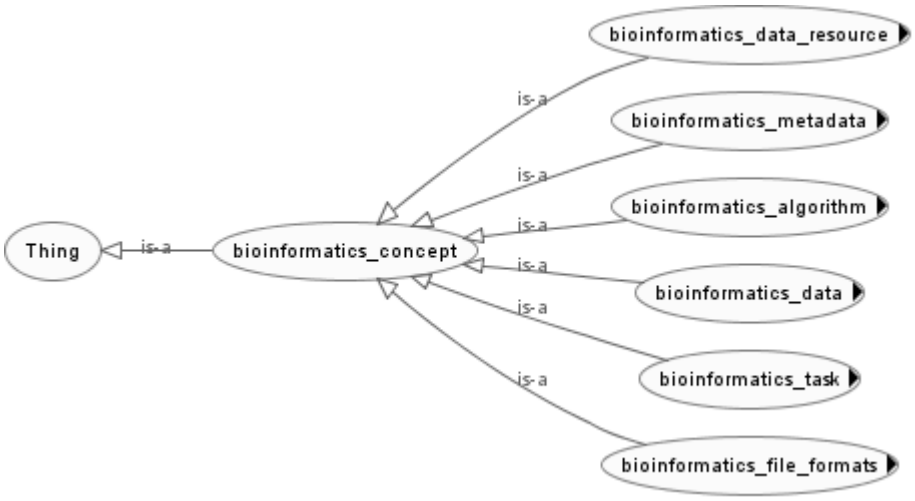

Figura 3.5: Ontologia *myGrid:* Classes Principais

A seguir, uma breve comparação entre a ontologia *myGrid* e a ontologia utilizada neste trabalho é apresentada. A Figura 3.5 apresenta uma visão parcial da hierarquia de termos/classes da ontologia *myGrid* e a Figura 3.6 apresenta esta mesma visão da

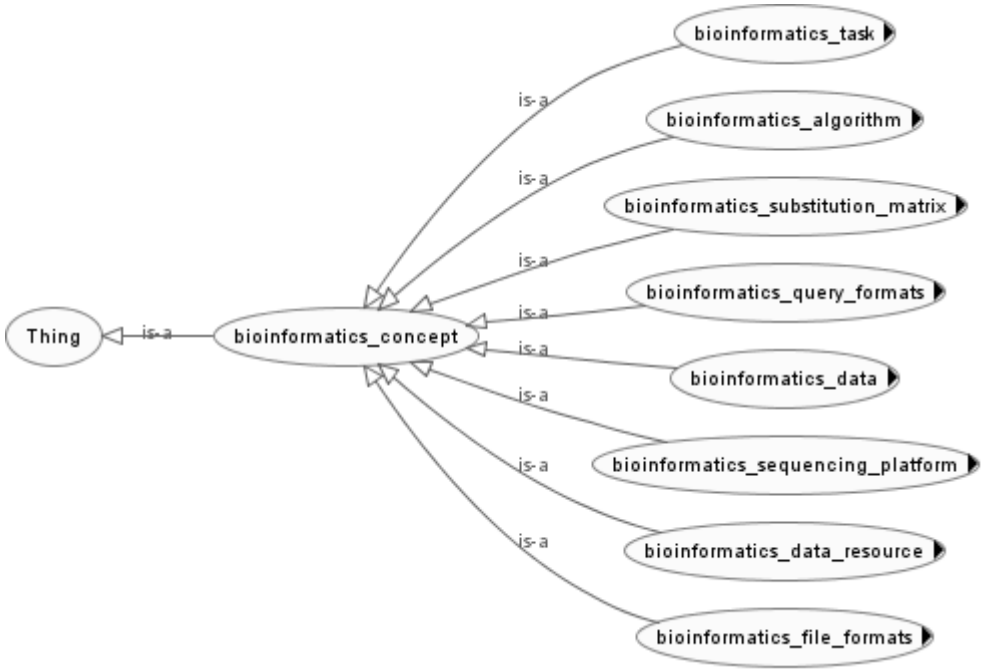

Figura 3.6: *Sequence Alingment Ontology:* Classes Principais

ontologia para alinhamento de sequências (*Sequence Alignment Ontology*) utilizada para sequenciamento/alinhamento genético na abordagem PL-Science.

De acordo com as Figuras 3.5 e 3.6, nota-se que a classe *bioinformatics_metadata* foi removida e as classes *bioinformatics_sequencing_ platform*, *bioinformatics_query_format*, *bioinformatics_substitution_matrix* foram adicionadas na *Sequence Alignment Ontology*. A classe *bioinformatics_metadata*, bem como suas subclasses, foram removidas por não serem necessárias de acordo com o objetivo da LPSC desenvolvida, e foram adicionadas novas classes de forma a possibilitar que as características e restrições do domínio escolhido pudessem ser contempladas. Por exemplo, a ontologia *myGrid* não apresentava nenhum conceito relacionado a plataformas de sequenciamento existentes, portanto, optou-se por incluir uma hierarquia de classes que abordasse esta questão, incluindo a classe *bioinformatics_sequencing_platform* e suas subclasses. O mesmo ocorreu com as demais classes que foram incluídas na ontologia, ou seja, por necessidade de especificar termos antes inexistentes e que eram necessários para o domínio escolhido.

Outra alteração significativa realizada na ontologia foi com relação às subclasses da classe *bioinformatics_taks*, conforme apresentado nas Figuras 3.7 e 3.8. Na *Sequence Alingment Ontology*, são descritas apenas as tarefas relacionadas ao sequenciamento/alinhamento genético. As demais tarefas que não estavam enquadradas neste subdomínio

foram removidas e aquelas que não estavam listadas na ontologia *myGrid* foram acrescentadas. Este mesmo procedimento foi realizado com relação à classe *bioinformatics_algorithm*, que, no caso da *Sequence Alingment Ontology*, englobará apenas os algoritmos necessários para o sequenciamento/alinhamento genético.

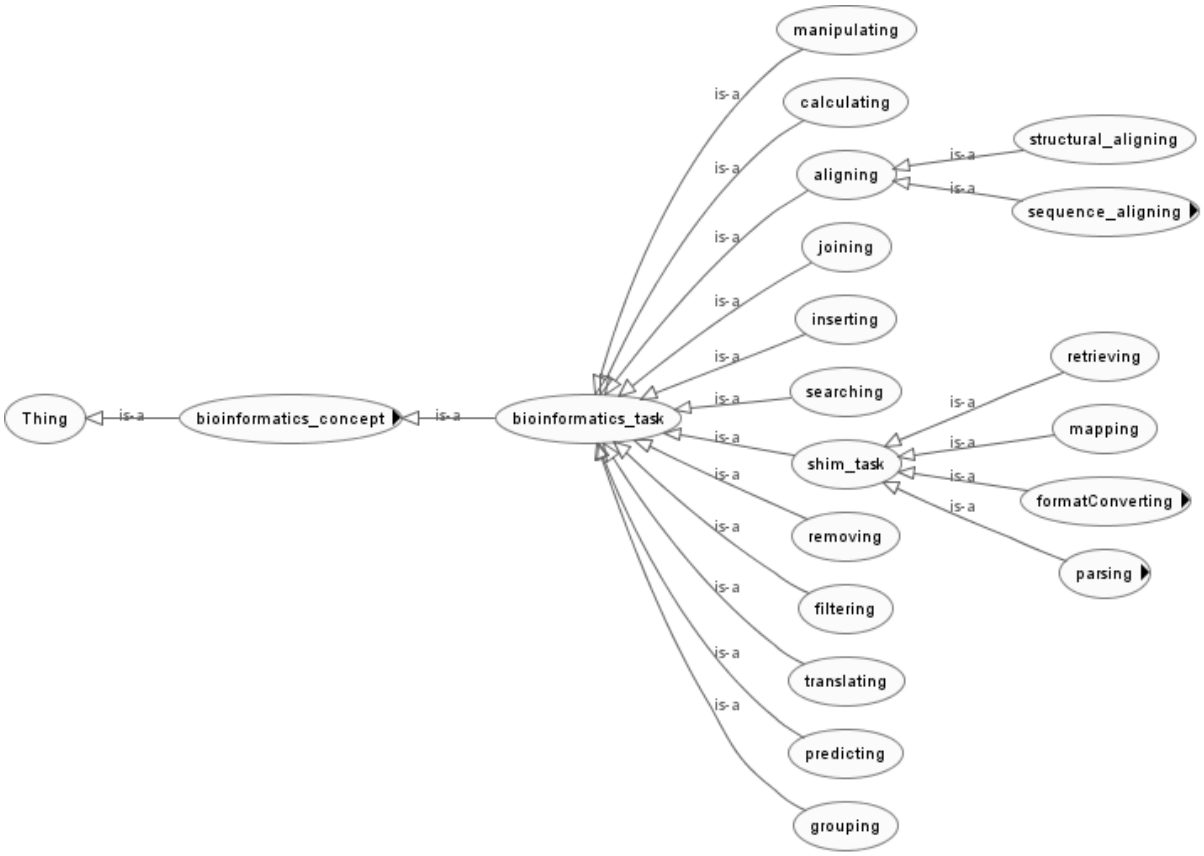

Figura 3.7: Ontologia *myGrid:* Classe *bioinformatics_task*

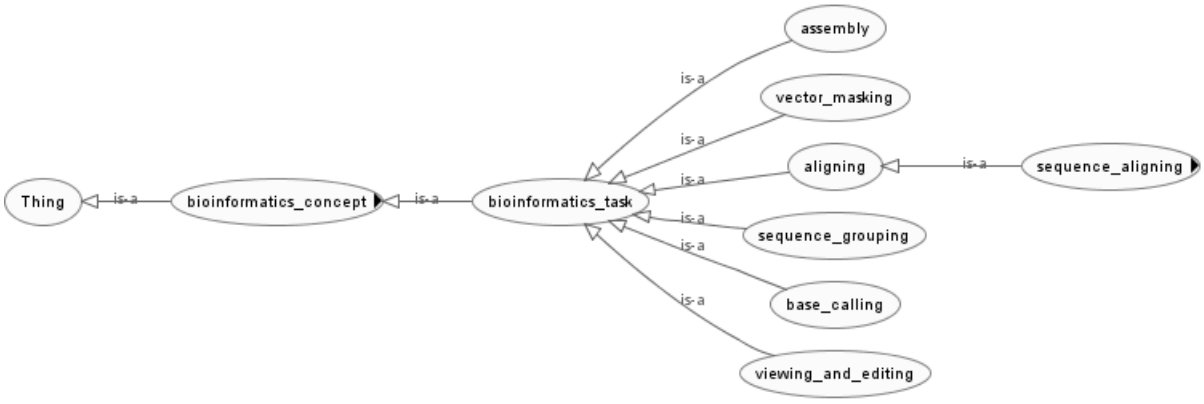

Figura 3.8: *Sequence Alignment Ontology:* Classe *bioinformatics_task*

Por fim, deve-se ressaltar que a definição de novas restrições na ontologia *myGrid* é o principal diferencial da *Sequence Alignment Ontology*, conforme apresentado na Figura

3.4 e nas Figuras 3.9 e 3.10.

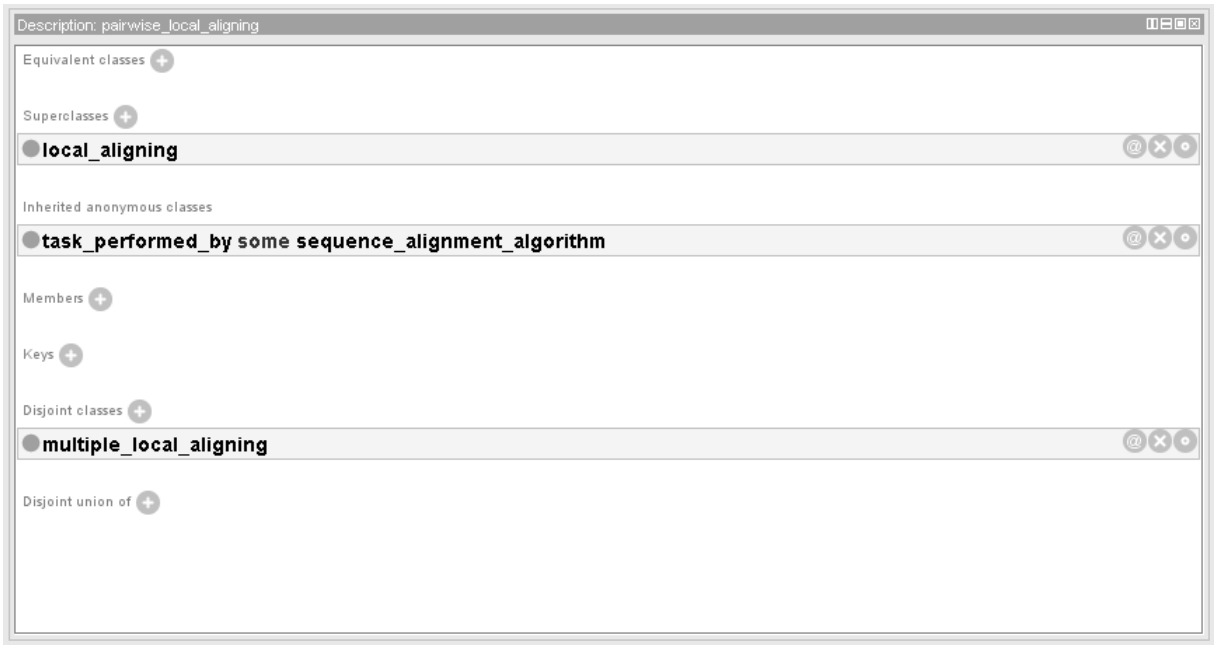

Figura 3.9: Ontologia *myGrid:* Restrições da Classe *pairwise_local_aligning*

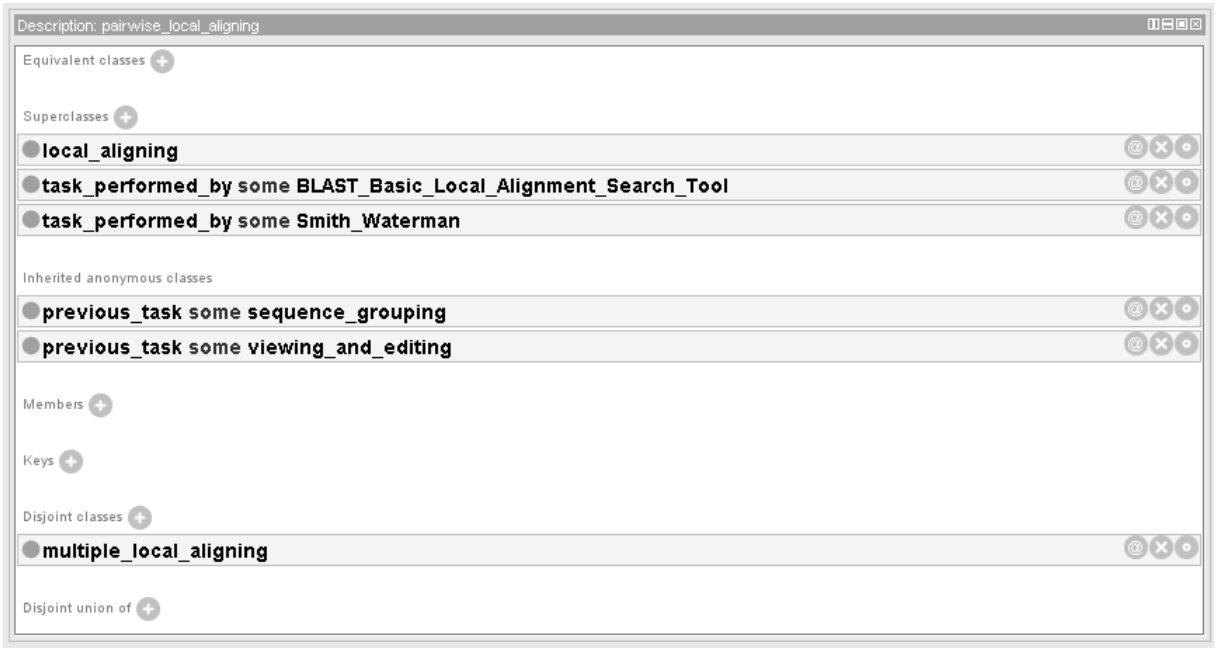

Figura 3.10: *Sequence Alignment Ontology:* Restrições da Classe *pairwise_local_aligning*

3.2.3 MAPEAMENTOS

Considerando a importância de auxiliar o cientista na tarefa de definição e implementação de novas aplicações, este trabalho propõe a utilização de modelos de *features* e ontologias,

que combinados, podem trazer maior expressividade semântica para a seleção de características desejáveis no desenvolvimento de *workflows* e/ou aplicações científicas no domínio de sequenciamento/alinhamento genético.

No intuito de estabelecer uma conexão entre o modelo de *features* e a ontologia anteriormente apresentados, criou-se um arquivo de mapeamento, especificado em XML. A Figura 3.11 ilustra a função deste mapeamento. O referido arquivo tem como objetivo principal estabelecer a correspondência existente entre as *features* do modelo de *features* e as classes da ontologia. Optou-se pela criação deste arquivo de mapeamento pois, nem sempre, as *features* do modelo estarão relacionadas a classes da ontologia de mesmo nome. Por este motivo, escolheu-se como forma de interligação entre estes modelos a especificação de um arquivo, que deverá ser criado manualmente, com o apoio do cientista do domínio. Com isso, busca-se alcançar uma maior corretude neste mapeamento e evitar a perda de dados (ligações entre os modelos), o que poderia afetar o desenvolvimento das aplicações científicas.

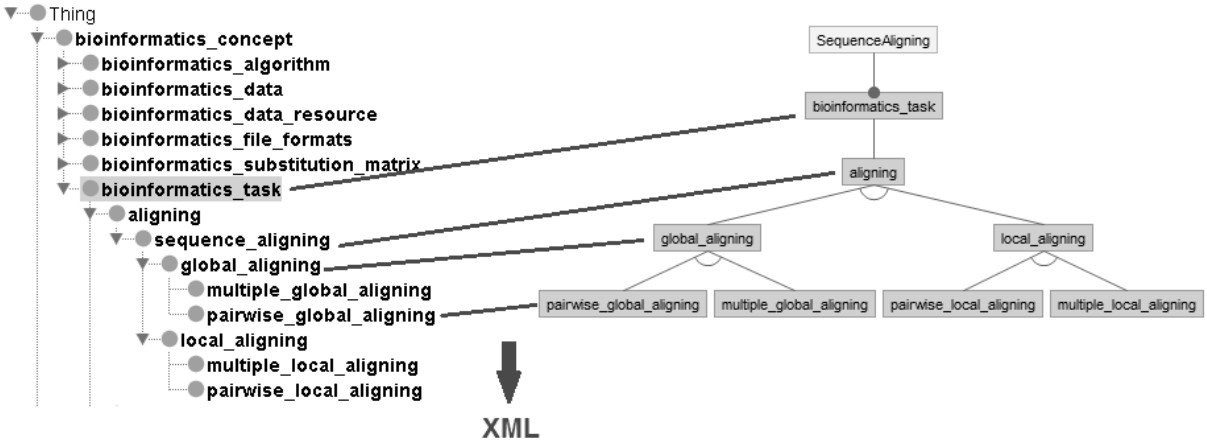

Figura 3.11: Função do Arquivo de Mapeamento Especificado em XML

Considerando o domínio de sequenciamento/alinhamento genético, um exemplo do conteúdo deste arquivo de mapeamento é apresentado na Figura 3.12. Nesta figura, a classe *TBLASTX* na ontologia corresponde à *feature* de mesmo nome, *TBLASTX*, no modelo de *features*. Outro exemplo é o mapeamento da classe *task_performed_by* e a *feature bioinformatics_algorithm*, como se pode visualizar no mapeamento com o ID 42 desta figura.

A manipulação dos três modelos apresentados anteriormente é coordenada pelo Gerente de Variabilidade apresentado na Seção 3.3.1.

```
−<SequenceAligningMapping>
        ...
 −<mapping id="41">
     <className>TBLASTX</className>
     <featureName>TBLASTX</featureName>
  </mapping>

 −<mapping id="42">
     <className>task_performed_by</className>
     <featureName>bioinformatics_algorithm</featureName>
  </mapping>
        ...
</SequenceAligningMapping>
```

Figura 3.12: Trecho do Arquivo de Mapeamento

3.2.4 MODELO ABSTRATO DE *WORKFLOW* CIENTÍFICO

Além da especificação do modelo de *features*, da ontologia e do mapeamento entre estes, a abordagem PL-Science requer a definição de um modelo abstrato de *workflow* para o domínio escolhido. A Figura 3.13 exibe o modelo de *workflow* utilizado para alinhamento/sequenciamento genético, o qual foi desenvolvido com base nas atividades descritas em (PROSDOCIMI et al., 2002) e com o auxílio de especialistas de domínio. Vale ressaltar que este modelo é apenas uma representação visual do *workflow* usado como base para a abordagem. Todas as atividades envolvidas neste modelo de *workflow* foram incluídas no modelo de *features* e as possibilidades de subfluxos que podem ser gerados a partir do modelo abstrato de *workflow* são descritas através da ontologia de domínio anteriormente apresentada.

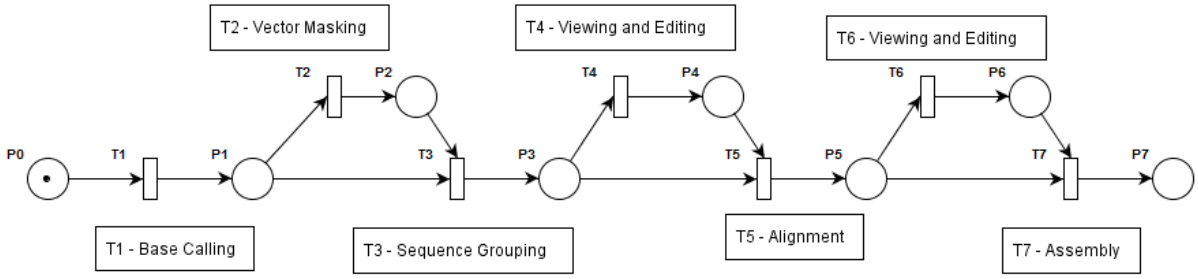

Figura 3.13: Modelo Abstrato de *Workflow* para Sequenciamento/Alinhamento Genético

De acordo com a Figura 3.13, foram identificadas sete atividades principais ou tarefas que compõem o modelo abstrato de *workflow* para alinhamento/sequenciamento gené-

tico, sendo elas: (T1) *Base Calling*, (T2) Mascaramento de Vetores, (T3) Agrupamento de Sequências, (T4) Visualização e Edição, (T5) Alinhamento, (T6) uma nova etapa de Visualização e Edição e (T7) Montagem. Uma breve descrição de tais atividades é apresentada no Capítulo 4, que trata da implementação da abordagem proposta.

3.3 ARQUITETURA DA SOLUÇÃO

A arquitetura da abordagem PL-Science, de acordo com a Figura 3.14, é dividida em duas camadas principais, a Camada Cliente (*Client Layer*), que engloba a interface Web da Aplicação e a Camada Principal (*Core Layer*). Nesta camada, merecem destaque os Gerentes da Linha de Produtos, de Variabilidade e Arquitetural (COSTA et al., 2012a).

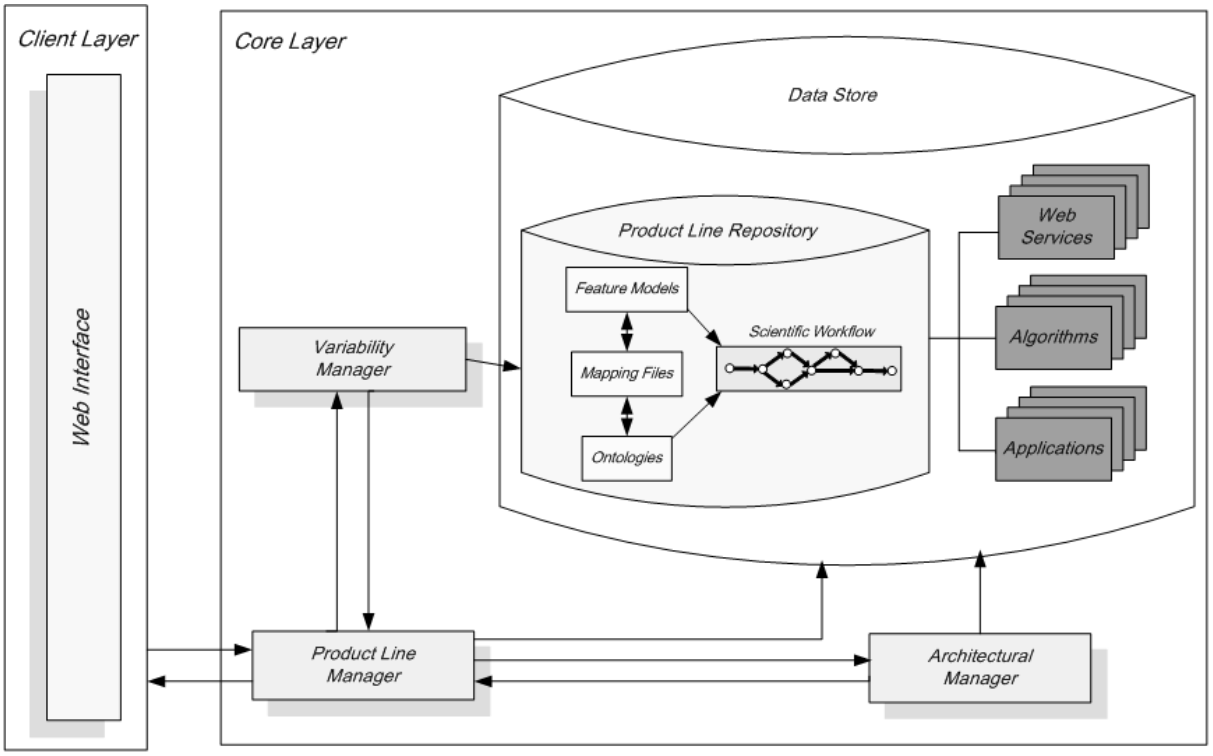

Figura 3.14: Visão Geral da Arquitetura PL-Science

Estes três gerentes interagem com o Repositório de Artefatos da LPSC, que inclui os componentes para a geração de *workflows* científicos da LPS proposta. Este repositório contém o modelo de *features* que descreve as variabilidades do domínio da LPSC e também uma ontologia, utilizada para descrever a semântica formal do domínio. Para conectar esses dois modelos há um arquivo de mapeamento, no formato XML, que mapeia os termos da ontologia e os termos do modelo de *features*.

3.3.1 GERENTE DE VARIABILIDADE (*VARIABILITY MANAGER*)

O Gerente de Variabilidade é responsável pelo acesso e manipulação dos modelos que representam as variabilidades na LPS. Além disso, como a abordagem proposta visa obter as vantagens do uso de ontologias e modelo de *features*, este gerente realiza a conexão entre estes dois modelos, através do arquivo de mapeamentos. Portanto, o controle das variabilidades existentes na LPSC é vinculado a estes três artefatos (ontologia, modelo de *features* e arquivo de mapeamento) e é realizado pelo Gerente de Variabilidade (*Variability Manager*), conforme Figura 3.15.

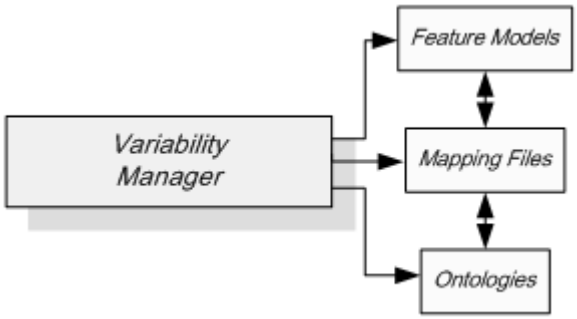

Figura 3.15: Artefatos Manipulados pelo Gerente de Variabilidade

O Gerente de Variabilidade proposto na arquitetura realiza os seguintes passos para gerenciar as variabilidades da LPSC:

- **Passo 1**: Conecta-se ao repositório da Linha de Produtos e carrega os modelos de *features*, a ontologia e o arquivo de mapeamento entre estes modelos.

- **Passo 2**: Captura as *features* (ou *subfeatures*) que foram selecionadas pelo usuário e que foram 'recebidas' através do Gerente da Linha de Produtos. Em seguida, acessa o arquivo XML de forma a obter a correspondência entre as *features* que foram selecionadas e os termos da ontologia (classes) que deverão ser analisados.

- **Passo 3**: Acessa a ontologia e obtém as restrições que estão relacionadas às *features* que foram selecionadas pelo usuário.

- **Passo 4**: Como as restrições expressas na ontologia referem-se a *features* que podem (ou devem) ser selecionadas para a geração do produto da LPSC, o gerente novamente traduz os termos contidos nas restrições em *features*, através do acesso ao arquivo de mapeamento.

- **Passo 5**: Finalmente, o novo conjunto de *features* é repassado ao Gerente da Linha de Produtos para ser apresentado ao usuário, de acordo com as restrições que foram encontradas.

- **Passo 6** - Repetição dos passos anteriores (do **Passo 2** ao **Passo 5**), ou seja, estes passos devem ser executados até que sejam estabelecidas todas as características necessárias para a geração de um produto final da LPS. Estes passos devem ser repetidos até que não haja mais restrições a serem atendidas de acordo com as características que foram informadas pelo usuário durante o processo de seleção de *features*. Vale enfatizar que, neste processo, o mecanismo de inferência é utilizado como uma forma de melhorar e/ou sugerir as melhores configurações de *features* ao usuário.

No Capítulo 4 serão apresentados alguns exemplos práticos de utilização do Gerente de Variabilidade no contexto de aplicações para o sequenciamento/alinhamento genético.

3.3.2 GERENTE DA LINHA DE PRODUTOS (*PRODUCT LINE MANAGER*)

Este gerente é responsável pela interação com o usuário da LPSC e também pela coordenação dos demais gerentes presentes na arquitetura. As funções relacionadas a este gerente é apresentada na Figura 3.16 e descrita de acordo com os três passos a seguir:

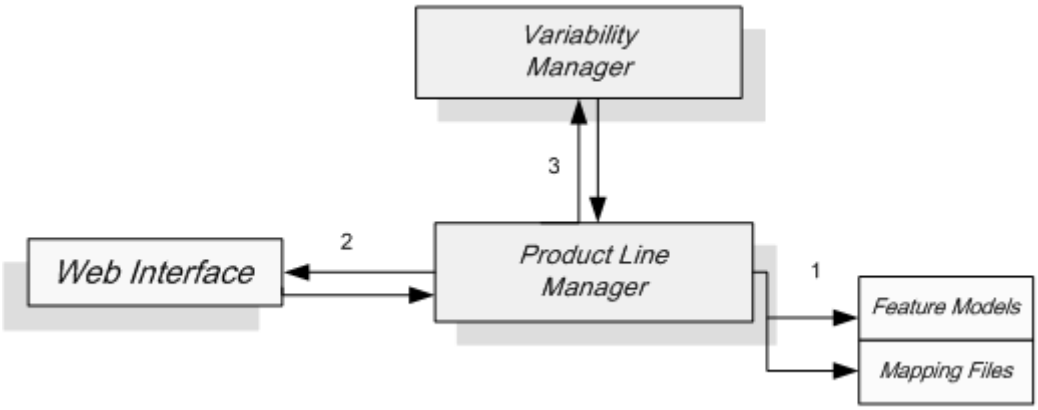

Figura 3.16: Funções iniciais do Gerente da Linha de Produtos

- **Passo 1**: Conectar-se ao repositório de artefatos e carregar o modelo de *features* e o arquivo de mapeamento da LPSC, ambos especificados usando XML;

- **Passo 2**: Exibir, com base no modelo de *features* da LPSC, a interface inicial para interação entre o usuário e a LPSC (Figura 3.17). Neste passo, as *features* marcadas como obrigatórias no modelo de *features* são exibidas em destaque (na cor laranja), ou seja, são mostradas em destaque somente as *features* incialmente obrigatórias do modelo de *features*, independente do *workflow* que será gerado através da LPSC. Estas *features* obrigatórias são apresentadas ao usuário com as suas *subfeatures* opcionais (se houverem), de forma que o usuário possa iniciar a configuração do *workflow*/aplicação a ser gerado(a).

- **Passo 3**: A seguir, o Gerente da Linha de Produto passa a comunicar-se diretamente com o Gerente de Variabilidades, de forma a auxiliar o usuário na escolha das *features* a serem selecionadas, de acordo com o *workflow*/aplicação final a ser gerado(a).

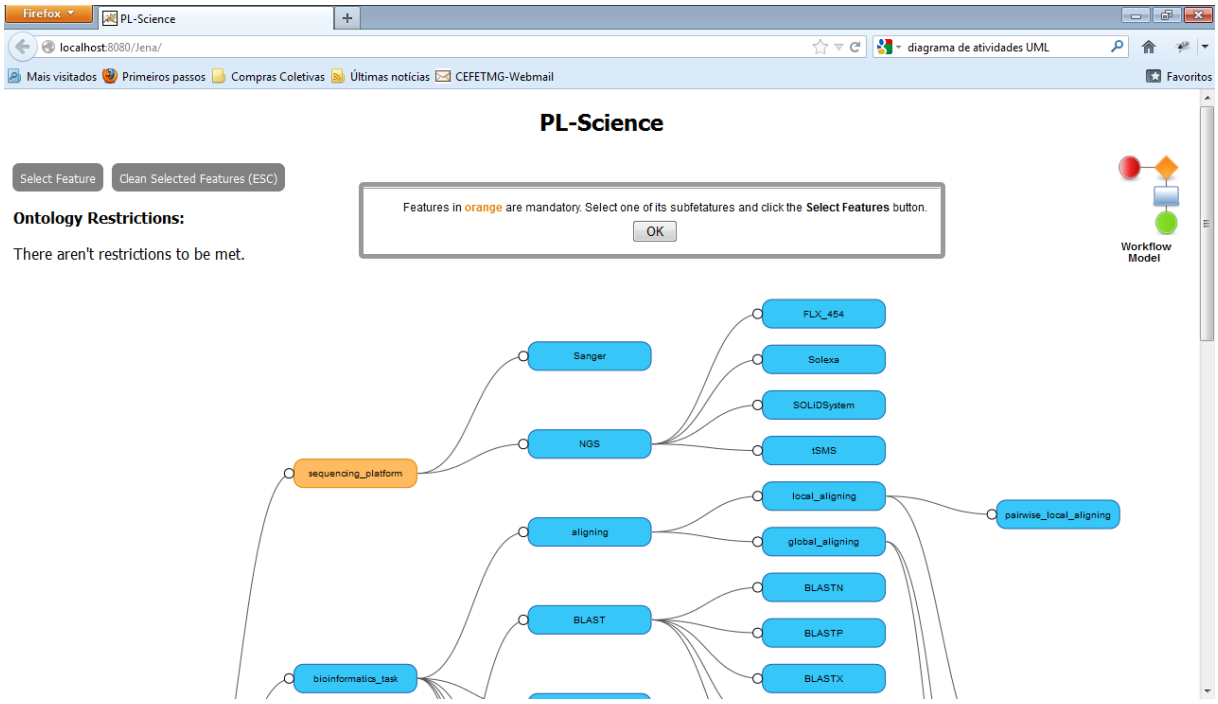

Figura 3.17: Tela inicial da Aplicação

Deve-se considerar que, a partir do **Passo 3**, o Gerente da Linha de Produtos passa a atuar como 'intermediário' entre o usuário da aplicação e o Gerente de Variabilidade. Parte-se do princípio de que este usuário irá selecionar as *features* existentes na LPSC e o Gerente de Variabilidade será responsável por analisar as escolhas deste usuário e instruí-lo neste processo. Desta forma, o gerente da LPSC intermedia a interação do usuário com o Gerente de Variabilidade.

Neste momento inicial, o Gerente da Linha de Produtos funciona como um controlador (*controller*) da aplicação, de acordo com o padrão de projeto *Model View Controller* (MVC) (HOFMEISTER et al., 2000).

Vale ressaltar que, no momento, este gerente tem apenas este papel de interação com o usuário e os outros gerentes da LP, porém, em uma versão futura da abordagem, ele deverá englobar tarefas que possibilitem a evolução da LPSC.

3.3.3 GERENTE ARQUITETURAL (*ARCHITECTURAL MANAGER*)

No contexto de LPS, em geral, é de grande importância a definição da arquitetura que será utilizada como base para a instanciação de aplicações em um determinado domínio. Ao contrário da arquitetura de software de produtos individuais, uma arquitetura de LPS deve englobar os requisitos de diferentes produtos que constituem a LPS, representando tanto os pontos comuns, como as variabilidades existentes na LPS (AHN; KANG, 2011).

Na arquitetura da abordagem PL-Science, este gerente possibilita a utilização de arquiteturas configuráveis na LPSC. Porém, no contexto deste trabalho, o detalhamento deste gerente será tratado como uma possibilidade de trabalho futuro. Embora não tenha sido implementado, este gerente foi incluído na arquitetura devido à importância para o aprimoramento da abordagem PL-Science. O gerente mencionado deve contemplar a identificação/definição dos elementos arquiteturais que irão compor a arquitetura da LPSC. Outra hipótese a ser validada é a possibilidade de facilitar esta identificação/definição através da utilização de modelos semânticos, tais como as ontologias.

3.4 METODOLOGIA

Considerando a arquitetura da PL-Science e sua utilização no contexto de uma LPSC, é importante o detalhamento de uma metodologia de utilização da mesma, onde se considere uma abordagem adequada para LPS e o contexto científico. Desta forma, foi proposta uma metodologia de alto nível com nove passos principais, divididos em duas fases: (1) **Desenvolvimento do Núcleo de Artefatos** ou **Engenharia de Domínio**, que descreve os artefatos produzidos na fase de engenharia de domínio e (2) **Desenvolvimento do Produto usando o Núcleo de Artefatos** ou **Engenharia de Aplicação**, que descreve a configuração dos produtos utilizando o núcleo de artefatos. Esta metodologia

será aplicada no desenvolvimento dos casos de uso apresentados no Capítulo 4.

- **Fase 1**: Desenvolvimento do Núcleo de Artefatos
  - **Passo 1**: Definição do escopo da linha de produtos.
  - **Passo 2**: Definição do modelo de *features* de acordo com as possibilidades de variação existentes no domínio escolhido.
  - **Passo 3**: Definição da ontologia de domínio, que contém as restrições a respeito do domínio escolhido que não podem ser expressas apenas através do modelo de *features*.
  - **Passo 4**: Realização do mapeamento entre os termos do modelo de *features* e a ontologia do domínio.
  - **Passo 5**: Definição de um modelo de *workflow* base, de acordo com o domínio escolhido.
  - **Passo 6**: Definição e conexão de outros artefatos (algoritmos, *Web Services*, entre outros) da LPSC que serão armazenados no repositório.
- **Fase 2**: Desenvolvimento do Produto usando o Núcleo de Artefatos
  - **Passo 1**: Seleção das *features* disponíveis no domínio escolhido, de acordo com o produto a ser desenvolvido (*workflow* científico). Esta seleção será baseada tanto no modelo de *features* e seu mapeamento para o modelo ontológico, quanto no(s) modelo(s) de *workflow(s)* disponível(is) para o domínio. Neste momento deve-se considerar o escopo da LPSC. Este passo é controlado pelo Gerente de Variabilidade.
  - **Passo 2**: Para cada tarefa do *workflow*, são analisadas as possibilidades de variação (definidas no **Passo 1**) e, em seguida, o usuário define qual o algoritmo, o *Web Service* ou a Aplicação que será 'instanciado' no *workflow* base.
  - **Passo 3**: Geração de um arquivo especificado em XML com o detalhamento das tarefas que deverão compor o *workflow* científico.

3.5 TRABALHOS RELACIONADOS

Existem na literatura alguns trabalhos que fazem uso de ontologias para a representação da variabilidade em LPS. Deve-se notar que na maioria destes trabalhos também são evidenciadas questões como deficiências ou 'não adequação' de modelos de *features* para expressar as variabilidades em um determinado domínio. Alguns dos trabalhos substituem o uso do modelo de *features* por ontologias, outros sugerem alterações no modelo de *features* de forma a aproximá-lo do formalismo de ontologias e outros ainda adotam o uso conjunto de ambos os modelos.

JOHANSEN et al. (2010) exploram modelos de *features* e ontologias, apresentando mecanismos de mapeamento que podem ser estabelecidos entre estes modelos. O trabalho discute a diferença entre esses modelos e apresenta cenários em que ontologias podem ser utilizadas para a modelagem de *features*. Além disso, JOHANSEN et al. (2010) propõem uma associação entre o modelo de *features* e a ontologia que pode ser utilizada para verificar a consistência entre estes modelos. Esta associação entre os modelos também pode ser usada para refatorar a hierarquia de um modelo de *features*. Embora esta dissertação aborde o uso de ontologias em conjunto com modelos de *features*, como em (JOHANSEN et al., 2010), o objetivo principal desta não é explorar os possíveis mecanismos de mapeamento que podem ser realizados entre estes modelos. Tem-se como propósito utilizar as vantagens oferecidas pelo uso conjunto de modelos de *features* e ontologias para, ao final, obter uma LPS para geração de *workflows* científicos em um domínio científico previamente estabelecido. Portanto, o diferencial desta dissertação é explorar o poder semântico provido pelo modelo ontológico em conjunto com a questão das variabilidades tratadas pelo modelo de *features*, visando facilitar a geração de produtos na LPSC proposta.

(ASIKAINEN et al., 2007) propõem o uso de uma ontologia de domínio para a modelagem das variabilidades em LPS chamada Kumbang. Esta ontologia unifica a modelagem de *features* e a modelagem arquitetural de famílias de produtos de software. A ontologia Kumbang foi descrita utilizando UML. Sendo assim, deve-se ressaltar que o poder da descrição semântica desta linguagem (UML) não é o mesmo que o de uma linguagem ontológica, como OWL. Restrições semânticas podem ser agregadas ao modelo UML através de OCL. No entanto, a linguagem OCL é de difícil entendimento e pouco conhecida da maioria dos desenvolvedores. Além disso, deve-se destacar os mecanismos de inferência que podem ser usados em conjunto com OWL. Por exemplo, em OWL podemos expressar

semânticas não contempladas pela UML, tais como: OWL permite que uma subclasse possa ser declarada usando o termo *subClassOf* e, adicionalmente, uma subclasse pode ser inferida a partir da definição de uma classe em termos de outras classes. Outro exemplo é a possibilidade de inferir, a partir das propriedades de um indivíduo, que ele é um membro de uma determinada classe do modelo de domínio.

Em (CZARNECKI et al., 2006), é apresentada uma comparação entre modelos de *features* e ontologias, explorando a relação existente entre os modelos. É proposta a adição de extensões ao modelo de *features*, tais como atributos para as *features*, de forma a aproximá-lo do formalismo existente nas ontologias. Diferentemente de CZARNECKI et al. (2006), este trabalho não tem como propósito melhorar o modelo de *features* adicionando extensões a este. Partiu-se da ideia de que a utilização em conjunto de dois modelos já definidos pode trazer vantagens com relação à adição de semântica no contexto de LPS. O modelo de *features* será utilizado para representar as variabilidades e a ontologia fornecerá as restrições que não se consegue representar através dos modelos de *features*. Além disso, na abordagem proposta nesta dissertação, o uso em conjunto de modelos de *features* e ontologias tem como objetivo a geração de uma LPS para *workflows* científicos.

Em (SILVA, 2011a), são apresentados modelos semânticos expressos em ontologias para representação de LPS. O autor afirma que esta forma de representação é importante para que possa existir um processo automatizado de construção, configuração e manutenção de uma LPS. Neste trabalho é descrita uma representação semântica de LP, a qual é utilizada como uma meta-representação de LPS. Através desta representação é possível descrever LPS usando ontologias. Relacionando este trabalho com a abordagem PL-Science, tem-se como diferencial que, na abordagem PL-Science, modelos de *features* são utilizados em conjunto com ontologias, ou seja, as vantagens de ambos os modelos podem ser utilizadas para a representação da variabilidade e da semântica em LPS.

Outro trabalho que possui pontos em comum com a abordagem PL-Science é apresentado em (FILHO et al., 2012). Neste trabalho são propostos métodos para adicionar informações de domínio, além de descrições de variabilidades, utilizando uma ontologia 'topo' que especifica conceitos genéricos e relações em uma LPS. Esta abordagem reusa o modelo de *features* da LPS, adicionando descrições semânticas à LPS, porém, não modifica a notação já existente do modelo de *features* utilizado. De forma a aumentar a

semântica de uma LPS, em um momento inicial, o modelo de *features* é automaticamente mapeado em uma ontologia. Tal abordagem é definida como uma "Linha de Produtos de Software Semântica" (*Semantic Software Product Line - SSPL*), na qual os artefatos reutilizáveis da LPS, e outros elementos organizacionais, encontram-se relacionados ao modelo de domínio da LPS, sendo que estes relacionamentos são expressos formalmente através de uma ou mais ontologias.

Como primeiro passo da abordagem *SSPL*, tem-se o mapeamento automático do modelo de *features* para um ontologia, a qual é chamada de esqueleto OWL. Este mapeamento consiste em obter um modelo de objetos (objetos em Java), diretamente, a partir de um modelo de *features*, ou seja, é realizada uma tradução do modelo de *features* para um modelo de objetos autocontidos. Os atributos dos objetos indexam as relações existentes entre eles. Esse modelo de objetos funciona como um modelo intermediário, a partir do qual é feita uma tradução para a linguagem OWL-DL, baseando-se em uma meta-ontologia. Assim, são gerados indivíduos, que são instâncias de conceitos da meta-ontologia proposta. Como segundo passo, é utilizada uma ontologia 'topo' que define a natureza das relações e conceitos que enriquecerão o esqueleto OWL. Por exemplo: (i) informações classificativas, que classificam as *features* de acordo com seus atributos em comum; (ii) relações de tipos de dados, definidos usando *DatatypeProperties* da OWL; e (iii) outras relações internas à LPS, que podem ser definidas usando *ObjectProperty* da OWL. Este 'enriquecimento' semântico do esqueleto OWL, gerado a partir do modelo de *features*, não é automatizado, ou seja, é realizado com a interferência do engenheiro de domínio da LPS. Por fim, como estudo de caso, a abordagem proposta neste trabalho é aplicada em LPS de guias móveis e sensíveis ao contexto. A partir dos resultados obtidos, foram analisados os benefícios da aplicação de tal abordagem, levando em consideração os aspectos de recuperação da informação, inferência e rastreabilidade destas.

Comparando a abordagem *SSPL* com a abordagem PL-Science, tem-se que a abordagem *SSPL* não foi desenvolvida para um domínio específico, ou seja, é genérica. Já a abordagem PL-Science, tem como foco o desenvolvimento de LPS para software científico. Portanto, vale ressaltar que neste domínio de software científico é exigido um grau maior de formalismo para especificação de restrições, principalmente se for considerado que determinados experimentos científicos podem conter processos que envolvem riscos, como vidas humanas, por exemplo. Na abordagem PL-Science, todo o processo de mapeamento

Tabela 3.1: Comparação dos Trabalhos Relacionados

Trabalho Relacionado	Modelo de *Features*	Ontologia	Mapeamento
(JOHANSEN et al., 2010)	Sim	Sim(OWL)	Regras de Mapeamento
(ASIKAINEN et al., 2007)	Não	Sim(UML)	-
(CZARNECKI et al., 2006)	Sim	Sim(OWL)	-
(SILVA, 2011a)	Não	Sim(OWL)	-
(FILHO et al., 2012)	Sim	Sim(OWL)	Modelo de Objetos
PL-Science	Sim	Sim(OWL)	Arquivo XML

entre modelos de *features* e ontologia deve ser completamente realizado com a interferência do engenheiro de domínio, de forma que não se perca nenhuma informação. Mais ainda, as restrições a serem estabelecidas entre as *features* são criadas com base no conhecimento deste engenheiro a respeito do domínio científico em questão. Nesta abordagem, não se tem um passo de tradução de modelos de *features* para uma ontologia. É criada uma relação entre estes modelos através de um arquivo de mapeamento, que interliga o modelo de *features* com classes de uma ontologia já existente. Ou seja, é realizado um mapeamento entre o modelo de *features* e uma ontologia já existente, sendo esta consagrada no domínio científico em questão, de forma que este mapeamento tira proveito do que cada modelo pode oferecer de melhor. O modelo de *features* fornece a representação da variabilidade da LPS e a ontologia é utilizada para a representação da semântica do domínio que não é coberta pela forma de representação da variabilidade de modelos de *features*. Assim, são utilizados dois modelos formais, sem a necessidade de modificar 'os formatos' de nenhum deles.

A Tabela 3.1 apresenta um resumo comparativo entre os cinco trabalhos relacionados apresentados e a abordagem PL-Science, considerando os seguintes aspectos: se utilizam modelos de *features* e ontologias (bem como a linguagem utilizada para a representação da ontologia) e se apresentam alguma forma de mapeamento entre modelo de *features* e ontologia.

Existem também outros trabalhos cujo foco é a especificação de experimentos científicos. Em (MATTOSO et al., 2010) e (OGASAWARA et al., 2009) é abordado o conceito de Linha de Experimento, tratando todas as etapas envolvidas em um experimento científico. Também há uma preocupação com a forma de especificação de experimentos científicos em um alto nível de abstração, conforme proposto em (OLIVEIRA et al., 2010).

Em (MATTOSO et al., 2010) é proposta uma abordagem para o gerenciamento de

experimentos em larga escala com base na coleta de proveniência, durante todas as fases do ciclo de vida do experimento, de forma auxiliar os cientistas a ter mais controle sobre o experimento científico. Em relação à abordagem PL-Science, esta tem como foco auxiliar o cientista na fase de composição/especificação de experimentos científicos, que são representados através de *workflows*. Nesta versão inicial da abordagem, as fases de execução e análise ainda não são tratadas.

Outro ponto que deve ser levado em consideração ao utilizar-se *workflows* científicos é a linguagem de representação destes fluxos de tarefas. Em (OLIVEIRA et al., 2010), é ressaltado que experimentos científicos quando modelados em um alto nível de abstração, independentemente das linguagens de especificação dos Sistemas de Gerenciamento de *Workflows* Científicos (SGWfC), apresentam diversas vantagens. Embora os atuais sistemas de gerenciamento de *workflows* científicos apoiem a execução dos *workflows*, eles apresentam limitações quanto à composição dos *workflows* quando se trata de usar diferentes níveis de abstrações, portanto, em (OGASAWARA et al., 2009) é apresentado o conceito de Linha de Experimento, que pode ser definida como um *workflow* conceitual que é capaz de derivar vários *workflows* no nível concreto. Linha de Experimento é tratada como uma abordagem sistemática para a composição de *workflows* científicos que representam um experimento *in silico*.

A abordagem PL-Science faz parte de um contexto maior de pesquisas em e-Science, que envolve a questão de composição de serviços, com a abordagem Composer-Science (SILVA et al., 2011) (SILVA et al., 2012), que propõe uma abordagem de composição de *workflows* científicos considerando questões semânticas e sintáticas, a partir de uma linguagem de descrição de alto nível.

3.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a arquitetura principal da abordagem PL-Science, bem como os principais modelos de domínio utilizados, como modelo de *features*, ontologia e arquivo de mapeamentos. Foram detalhados também, os gerentes da arquitetura, sendo eles os principais responsáveis pelo funcionamento e manipulação dos artefatos da LPS de forma a atingir os objetivos da mesma. A metodologia utilizada pela abordagem é especificada e também os trabalhos relacionados à proposta deste trabalho foram apresentados e comparados.

4 IMPLEMENTAÇÃO DA ABORDAGEM PL-SCIENCE

Este capítulo tem como objetivo apresentar a implementação da abordagem PL-Science, de acordo com o que foi descrito no Capítulo 3, no domínio da Bioinformática (sequenciamento/alinhamento genético). Este capítulo está organizado da seguinte forma: a Seção 4.1 apresenta uma breve introdução sobre a implementação da abordagem e a Seção 4.2 descreve o domínio de sequenciamento/alinhamento genético, de forma a facilitar o entendimento do cenário em que foi implementada a abordagem. São apresentadas as atividades de sequenciamento e alinhamento, bem como todas as outras atividades que compõem o modelo abstrato de *workflow*, brevemente apresentado na Seção 3.2.4. A Seção 4.3 apresenta, de fato, a aplicação da abordagem PL-Science de acordo com os modelos de domínio desenvolvidos (descritos na Seção 3.2) e com a metodologia proposta na Seção 3.4, que considera duas fases principais: (1) Desenvolvimento do Núcleo de Artefatos e (2) Desenvolvimento do Produto. Ao final desta seção, são apresentados os produtos gerados a partir da LPSC criada. Por fim, a Seção 4.4 apresenta as considerações finais deste capítulo.

4.1 INTRODUÇÃO

Com o objetivo de verificar a aplicabilidade da abordagem PL-Science, foi especificado um cenário de uso no domínio da Bioinformática, tendo como objetivo final a geração de *workflows* (ou aplicações isoladas) destinados ao processo de sequenciamento/alinhamento genético. Os *workflows* ou aplicações isoladas geradas são apresentados como produtos da LPSC proposta, na Seção 4.3. É importante ressaltar que, apesar de os modelos apresentados no Capítulo 3 e também neste capítulo - modelo de *features*, ontologia, arquivo de mapeamento entre ontologia e modelo de *features* e modelo abstrato de *workflow* - serem específicos do domínio de sequenciamento/alinhamento genético, a abordagem pode ser aplicada em outros domínios cientifícos, desde que modelos específicos sejam criados na fase de Engenharia de Domínio.

4.2 DOMÍNIO DE SEQUENCIAMENTO/ALINHAMENTO GENÉTICO

Uma das áreas de pesquisa que mais cresce no meio científico é a área da Bioinformática, que tem como objetivo auxiliar os pesquisadores em Biologia a manipular, desenvolver e melhorar os bancos de dados de Biologia Molecular e outras ferramentas computacionais para obter, organizar e interpretar os dados de experimentos coletados em laboratórios (PROSDOCIMI et al., 2002).

De acordo com os conceitos da Biologia, DNA (*deoxyribonucleic acid* - ácido desoxirribonucleico) e RNA (*ribonucleic acid* - ácido ribonucleico) são cadeias de polímeros compostas por uma categoria de substâncias químicas similares. As unidades individuais destas cadeias são denominadas nucleotídeos ou bases. O DNA possui em sua composição os nucleotídeos A (adenina), T (timina), G (guanina) e C (citosina), enquanto o RNA substitui o T pelo U (uracila), além de possuir os outros três nucleotídeos (DARNELL J. E., 1986). Estes dois conceitos são objeto de estudo da subárea de pesquisa em sequenciamento/alinhamento genético.

4.2.1 SEQUENCIAMENTO

O estudo do material genético de um organismo é possível através do processo de sequenciamento, que determina a sequência de nucleotídeos deste material. Um dos métodos de sequenciamento mais utilizados é o de Sanger (SANGER et al., 1977). Porém, existem também as tecnologias de sequenciamento de nova geração (NGS - *Next Generation Sequencing*), que começaram a ser comercializadas no ano de 2005 (CARVALHO; SILVA, 2010).

Resumidamente, o processo de sequenciamento se inicia com a quebra do DNA em diversos fragmentos aleatórios, que serão 'lidos' por uma máquina de sequenciamento (ou sequenciador) de DNA. O arquivo de saída do sequenciador é um cromatograma que contém a sequência e a qualidade das bases que foram obtidas. Estes cromatogramas consistem de quatro curvas de cores diferentes, sendo que cada uma das curvas representa uma das quatro bases da sequência gerada. Para converter estes cromatogramas em uma sequência de bases deve-se utilizar um programa específico para esta tarefa. Estes programas tem como função principal analisar o cromatograma gerado e fornecer uma nota para cada base, baseado em sua qualidade. Como exemplos destes programas podem ser cita-

dos: Abiview (EMBOSS, 2013), Chromas (TECHNELYSIUM, 2013) e Phred (EWING et al., 1998).

4.2.2 ALINHAMENTO

Há um princípio biológico que diz que: *se duas sequências (nucleotídicas ou protéicas) são similares, então, é razoável supor que suas funções também sejam similares.* Portanto, o primeiro passo é verificar se outros pesquisadores já estudaram sequências similares às novas sequências que foram obtidas. Assim, os alinhamentos entre sequências oferecem um ótimo meio de comparar sequências relacionadas (SILVA, 2010).

O alinhamento é uma etapa que ocorre posteriormente ao sequenciamento, ou seja, após o sequenciamento, é necessário identificar e agregar informações à sequência gerada. Uma forma de fazer isto é através da comparação da sequência gerada com sequências previamente conhecidas. Para isso, realiza-se o alinhamento da sequência gerada com um banco de dados de sequências já conhecidas/estudadas.

O conceito básico para a seleção de uma boa sequência de alinhamento é simples: duas sequências são combinadas aleatoriamente e a semelhança entre estas é avaliada e pontuada. Em seguida, uma sequência é movida em relação a outra e a combinação é pontuada novamente, até que seja obtida a melhor pontuação de alinhamento entre as sequências envolvidas.

4.2.3 MODELO ABSTRATO DE *WORKFLOW* PARA SEQUENCIAMENTO/ALINHAMENTO GENÉTICO

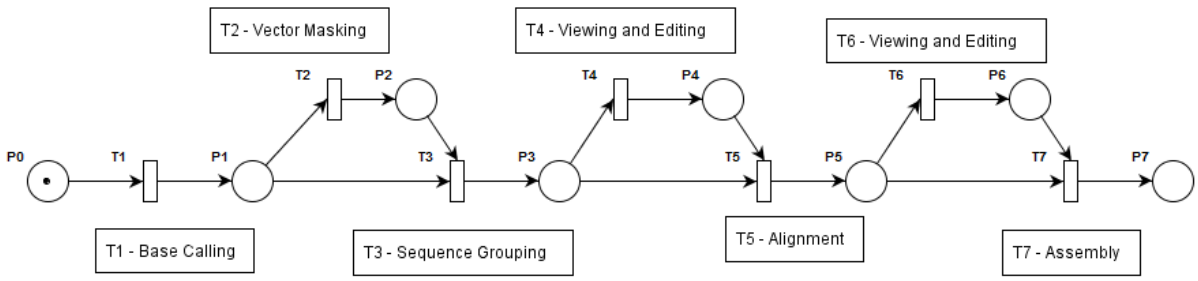

Figura 4.1: Modelo Abstrato de *Workflow* para Sequenciamento/Alinhamento Genético

De acordo com o domínio de alinhamento/sequenciamento genético, além das atividades de sequenciamento e alinhamento, outras atividades devem ser incluídas entre

(ou após) estas duas atividades principais. Considerando o contexto de um experimento científico neste domínio, estas outras atividades devem ser realizadas de acordo com um modelo de *workflow*, como o exibido na Figura 4.1. Uma breve descrição destas atividades é apresentada a seguir.

- *Base Calling* (T1)

  O base calling engloba o processo de leitura dos dados gerados por um sequenciador genético e o processo de identificação da sequência de DNA gerada, atribuindo um valor de qualidade para cada posição nucleotídica identificada. Geralmente, cada sequenciador possui seu programa de base calling específico (PROSDOCIMI et al., 2002). Ou seja, este processo tem por objetivo transformar os dados provenientes de um sequenciador em cadeias de nucleotídeos, também chamadas de *reads* (STEIN, 2002). Quando as sequências biológicas possuem um sequenciamento pelo método de Sanger, o programa mais utilizado nesta etapa é o Phred (PROSDOCIMI et al., 2002). Já os programas de *base calling* para sistemas NGS são muito sensíveis às tecnologias de sequenciamento de nova geração. Por exemplo, os arquivos de "imagens-primas" produzidos pela maioria dos sistemas de NGS são enormes, preenchendo rapidamente a capacidade de armazenamento em disco. Portanto, algumas máquinas, tais como a HiSeq da Illumina, não tornam os dados das "imagens-primas" disponíveis; a imagem é apenas processada e o arquivo da imagem é eliminado logo após o processamento (STEIN, 2002).

  No caso das técnicas de sequenciamento de nova geração, geralmente, os programas que fazem *base calling* são fornecidos pelas próprias empresas fabricantes dos sequenciadores, já que estes programas estão de acordo com a técnica de sequenciamento do fabricante (PROSDOCIMI et al., 2002). A principal desvantagem desta situação é que cada um dos programas de *base calling* oferecidos pelos fabricantes das plataformas NGS produzem diferentes *scores* ou pontuações de qualidade para as bases geradas que indicam a probabilidade de erros da sequência. Porém, estas pontuações não são comparáveis entre os diferentes fornecedores ou entre o software do fornecedor e o Phred. Uma pontuação de qualidade de 25 em um *base calling* feito pela plataforma SOLiD não indica o mesmo nível de confiança, como a mesma pontuação, em um *base calling* do HiSeq Illumina 2000 (STEIN, 2002).

- Mascaramento de Vetores (T2)

  Após o *base calling*, ou seja, após obter o arquivo com as notas de cada uma das base da sequência, pode-se realizar uma busca de regiões na sequência produzida que devem ser retiradas ou mascaradas, já que estas regiões não representam o DNA que se deseja analisar. Estas regiões são denominadas 'regiões contaminantes' (PROSDOCIMI et al., 2002). Esta fase nem sempre é realizada. Na maioria das vezes, ela é feita quando há a construção de uma biblioteca, pois os fragmentos de DNA são enxertados em um plasmídeo (vetores) para clonagem. Portanto, é necessário o mascaramento dos nucleotídeos do vetor, para que sejam estudados apenas os nucleotídeos de interesse. Como exemplo de programa que realiza esta tarefa de mascaramento de vetores, pode-se citar o Cross_match (CROSS_MATCH, 2013).

- Agrupamento de Sequências (T3)

  A tarefa de agrupamento de sequências (*sequence_grouping*) também é conhecida como "clusterização". Durante esta etapa, os fragmentos do DNA (*reads*), advindos do sequenciamento, são agrupados formando um *contig* (pequenos fragmentos de DNA sequenciados em sequências maiores). Os programas mais utilizados são o Phrap (PHRAP, 2013) e o CAP3 (CAP3, 2013), para o sequenciamento de Sanger (PROSDOCIMI et al., 2002). São usados Velvet (VELVET, 2013) e MIRA (MIRA, 2013) para NGS.

- Visualização e Edição (T4 e T6)

  Esta fase pode ser considerada como opcional, já que se trata de uma inspeção manual para verificação da montagem ou agrupamento das sequências. Portanto, se o número de sequências analisadas é relativamente pequeno, vale à pena passar por esta fase. Porém, ao trabalhar-se em escala genômica ou com muitas sequências, este passo de visualização e edição torna-se inviável e, por este motivo, na maioria das vezes, confia-se no resultado de programas que analisam os resultados obtidos na fase de agrupamento e atribuem valores de qualidade às sequências geradas. Entre os programas que possibilitam estas operações encontram-se o Phrapview e o Consed (PROSDOCIMI et al., 2002).

- Alinhamento (T5)

No contexto da Bioinformática, uma forma de fazer comparações entre DNA, RNA ou sequências de proteínas é através do processo de alinhamento de sequências (PROSDOCIMI et al., 2002), que consiste, basicamente, em colocar uma sequência sobre a outra e compará-las, de forma a especificar as partes comuns das sequências.

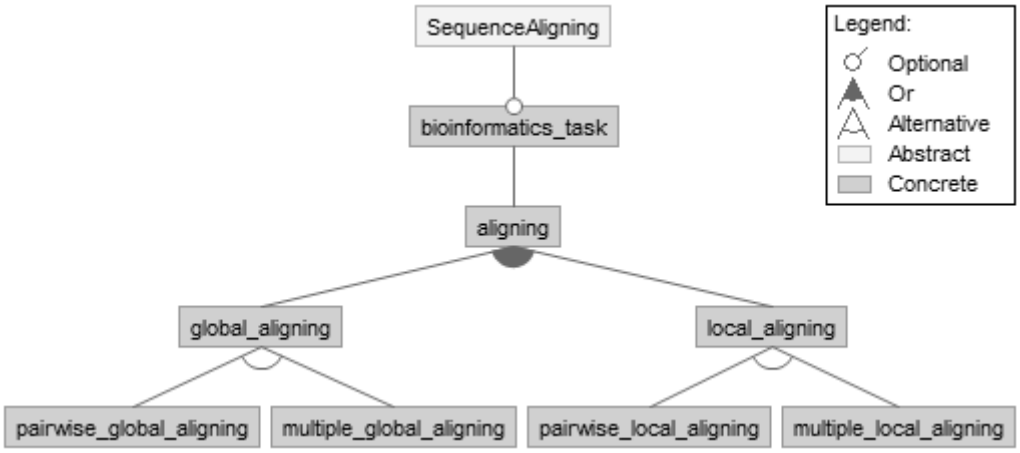

Figura 4.2: Tipos de Alinhamento

A escolha do algoritmo de alinhamento depende criticamente do tipo de alinhamento a ser realizado (STEIN, 2002). O alinhamento entre duas ou mais sequências pode ser dividido em local e global, conforme exibido na Figura 4.2.

- Alinhamento Local (*local_aligning*): realiza a comparação apenas de fragmentos das sequências envolvidas no processo de alinhamento;
- Alinhamento Global (*global_aligning*): realiza o alinhamento completo das sequências envolvidas no processo de alinhamento.

Além disso, o grau de semelhança entre as sequências pode ser estipulado para alinhamentos entre duas ou mais sequências. Assim, tanto o Alinhamento Global como o Alinhamento Local podem ser divididos em Duplo e Múltiplo.

- Alinhamento Duplo (*pairwise_aligning*): apenas duas sequências são envolvidas no processo de alinhamento;
- Alinhamento Múltiplo (*multiple_aligning*): várias sequências (ou mais do que duas sequências) são envolvidas no processo de alinhamento.

- Montagem (T7)

Outra tarefa que pode ser executada é a montagem (*assembly*) das sequências após o alinhamento. Esta tarefa ocorre quando não há sequências que possam ser usadas como referências, ou seja, quando não existem sequências produzidas a partir de um genoma/transcriptoma da mesma espécie da amostra ou de uma espécie relacionada que podem ser usadas como referências. Este tipo de sequenciamento, conhecido por sequenciamento *de novo*, exigirá uma montagem das sequências, utilizando apenas os dados obtidos desse alinhamento entre as sequências geradas.

4.3 APLICAÇÃO DA ABORDAGEM PL-SCIENCE

Nesta seção são apresentadas as etapas realizadas para a aplicação da abordagem PL-Science, considerando o domínio de alinhamento/sequenciamento genético, de acordo com o detalhamento do domínio de sequenciamento/alinhamento e com a metodologia proposta na Seção 3.4. São detalhadas as etapas para a aplicação da abordagem tanto para a especificação do núcleo de artefatos, através de uma abordagem de engenharia de domínio, bem como para o detalhamento da especificação de três produtos da linha, caracterizando o processo de engenharia de produto.

- **Fase 1**: Desenvolvimento do Núcleo de Artefatos (Etapa de Engenharia de Domínio)

  Considerando a **Fase 1**, detalhada no Capítulo 3, é descrito, a seguir, cada passo para o desenvolvimento do núcleo de artefatos no domínio de alinhamento/sequenciamento genético.

  - **Passo 1**: Definição do escopo da linha de produto

    * O escopo da LPS está relacionado à geração de *workflows* científicos que englobam as atividades envolvidas no processo de sequenciamento/alinhamento genético, sendo elas: *base calling*, mascaramento de vetores (*vector masking*), agrupamento de sequências (*sequence grouping*), visualização e edição (*visualization and edition*), alinhamento (*alignment*) e montagem (*assembly*), conforme descrito no modelo abstrato de *workflow*, na Seção 4.2.3.

  - **Passo 2**: Definição do modelo de *features* de acordo com as possibilidades de variação existentes no domínio escolhido.

    * Nesta etapa, o modelo de *features* foi definido. As Figuras 2.1, 2.2, 3.2 e 3.3 apresentam partes do modelo criado. O modelo de *features* completo,

especificado em xml, pode ser encontrado em: `http://gabriellacastro.com.br/PL-Science/SequenceAligningFeatureModel.xml`.

– **Passo 3**: Definição da ontologia de domínio, contendo as restrições a respeito do domínio escolhido.

  * A ontologia de domínio utilizada para a implementação da abordagem, conforme apresentado na Seção 3.2.2 foi derivada da ontologia *myGrid* (WOLSTENCROFT et al., 2007). A ontologia criada neste passo, chamada *Sequence Alignment Ontology*, pode ser encontrada em: `http://gabriellacastro.com.br/PL-Science/SequenceAligningOntology.owl`.

– **Passo 4**: Realização do mapeamento entre os termos do modelo de *features* e a ontologia do domínio.

  * Neste passo, o arquivo de mapeamento que relaciona os termos do modelo de *features* com os termos da ontologia (e vice-versa) foi criado. O arquivo de mapeamento completo está disponível no seguinte link: `http://gabriellacastro.com.br/PL-Science/SequenceAligningMapping.xml`.

– **Passo 5**: Definição de um modelo de *workflow* base, de acordo com o domínio escolhido.

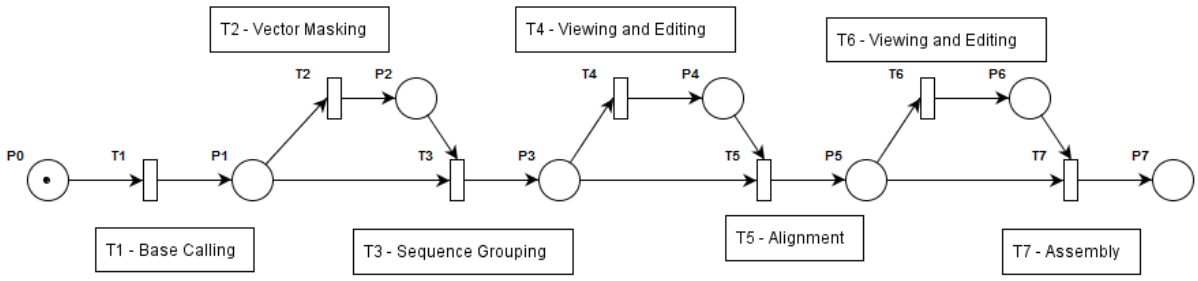

Figura 4.3: Modelo Abstrato de *Workflow* Criado

  * O modelo de *workflow* criado neste passo (Figura 4.3) apresenta o encadeamento de todas as tarefas que podem estar envolvidas no processo de sequenciamento/alinhamento genético, sendo estas descritas na Seção 4.2.3. Este *workflow* foi criado com auxílio de especialistas em Bioinformática, de forma a especificar um modelo genérico do que ocorre nos experimentos relacionados ao domínio escolhido. Embora o encadeamento das tarefas esteja esquematizado na Figura 4.3, a abordagem PL-Science captura este encadeamento através da

ontologia do domínio, que possui todas as tarefas cadastradas como subclasses de *bioinformatics_task* (Figura 4.4). Estas tarefas (que são representadas na ontologia como subclasses de *bioinformatics_task*) são relacionadas entre si através de restrições na ontologia que utilizam a *ObjetctProperty previous_task*. Por exemplo, a tarefa T3 (agrupamento de sequências) possui como *previous_task* tanto *base_calling* como *vector_masking*, conforme exibido na Figura 4.5. Desta forma, a única tarefa que não possui nenhuma *previous_task* relacionada a ela é a primeira, ou seja, o *base_calling*.

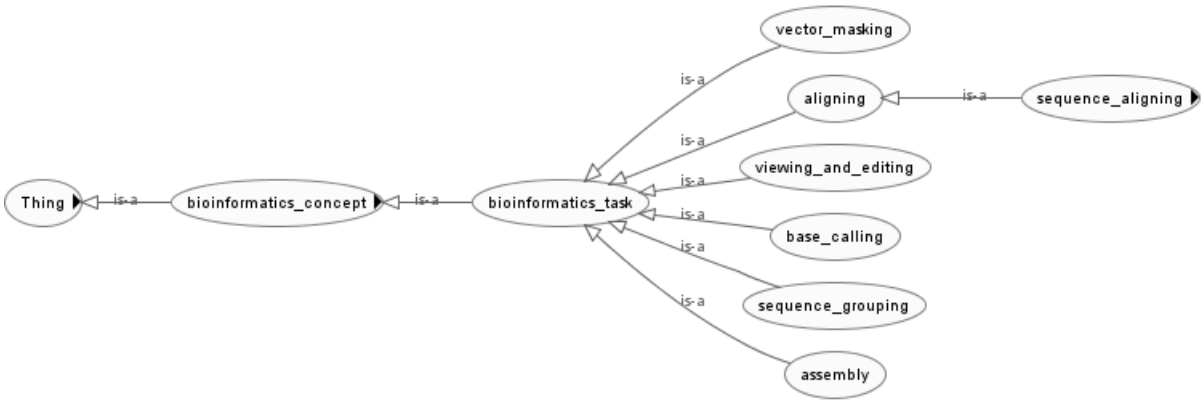

Figura 4.4: Tarefas do Modelo Abstrato de *Workflow*

```
<owl:Restriction>
    <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#previous_task"/>
    <owl:someValuesFrom rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#vector_masking"/>
</owl:Restriction>

<owl:Restriction>
    <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#previous_task"/>
    <owl:someValuesFrom rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#base_calling"/>
</owl:Restriction>
```

Figura 4.5: Restrições da Classe *sequence_grouping*

– **Passo 6**: Definição e conexão de outros artefatos (algoritmos, *Web Services*, entre outros) da LPSC que serão armazenados no repositório.

  * Nesta etapa, *Web Services*, algoritmos e aplicações foram catalogados como artefatos da LPSC. Estes artefatos serão utilizados em uma etapa posterior para compor o *workflow* científico a ser gerado. Como exemplos de *Web Services* que foram catalogados, pode-se citar os serviços catalogados no BioCatalogue (BHAGAT et al., 2010), como o *runPhrapService* (`http://www.biocatalogue.org/services/1567`) e o *INB-dev: genome.imim.es: runPhrap* (`http://www.biocatalogue.org/services/2268`).

- **Fase 2**: Desenvolvimento do Produto (Etapa de Engenharia da Aplicação)

  Baseado na **Fase 2** da metodologia apresentada no Capítulo 3 deste trabalho, é apresentada a especificação de *workflows* científicos ou aplicações isoladas no domínio de alinhamento/sequenciamento genético. É apresentado a seguir a especificação de três produtos gerados a partir da LPSC de Alinhamento/Sequenciamento Genético. O Produto 1 detalha a criação de um *workflow* que envolve as três tarefas iniciais do Modelo Abstrato de *Workflow*, que auxilia o cientista na realização da leitura, mascaramento e agrupamento das sequências advindas de um sequenciador genético; o Produto 2 descreve a criação de uma aplicação isolada para o alinhamento de múltiplas sequências; e, finalmente, o Produto 3 apresenta um *workflow* que trata desde a obtenção dos dados de uma sequência através de um sequenciador até a comparação desta sequência obtida com bases de sequências já existentes, através do alinhamento.

4.3.1 PRODUTO 1 - PIPELINE PHREDPHRAP

Nesta primeira especificação de um produto a partir da LPSC no domínio de sequenciamento/alinhamento genético, tem-se como objetivo a criação de um *workflow* baseado em (ARBEX, 2009), que consiste em:

- Ler os dados gerados por um sequenciador genético e identificar a sequência de DNA gerada, atribuindo valores de qualidade para cada posição nucleotídica identificada;
- Realizar uma busca de regiões na sequência produzida que devem ser retiradas ou mascaradas;
- Agrupar os *reads* advindos do processo de leitura dos dados formando em sequências maiores.

A execução dos passos acima possibilita ao cientista a realização de análises, montagens, e editoração de sequências. Assim, este *workflow* envolve as tarefas T1 (*base calling*), T2 (*vector masking*) e T3 (*sequence grouping*) do modelo abstrato de *workflow* do domínio, conforme exibido na Figura 4.6.

Através da metodologia da abordagem, os passos seguidos são detalhados a seguir:

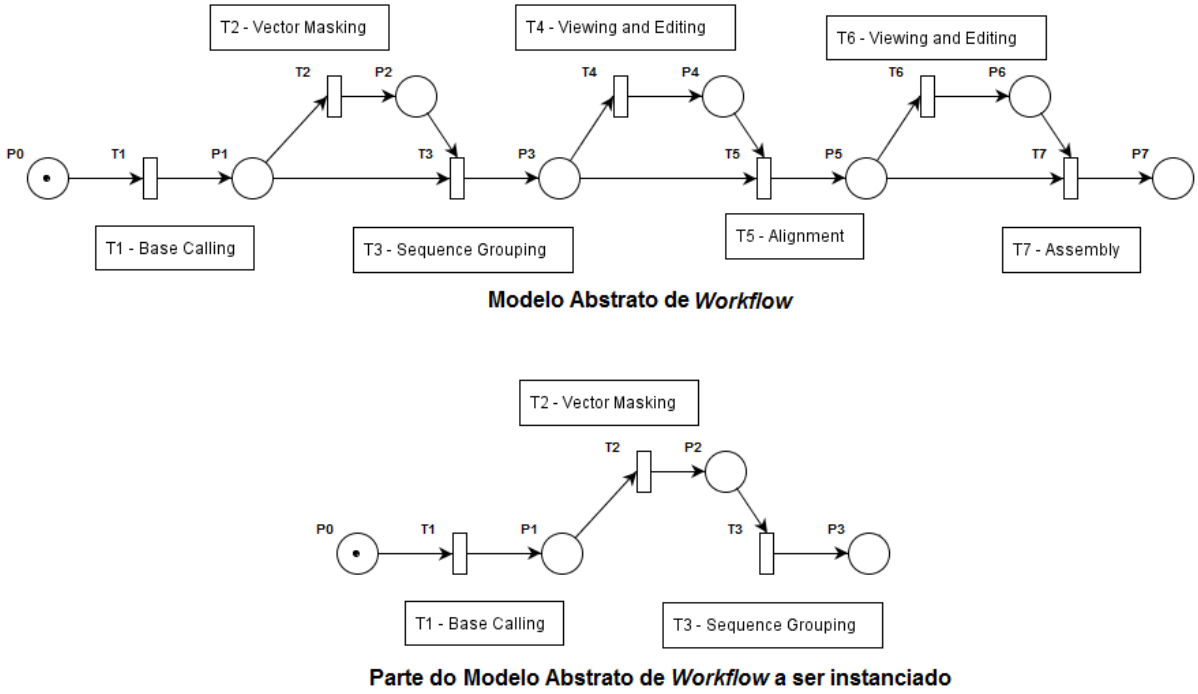

Figura 4.6: Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 1)

– **Passo 1**: As *features* inicialmente obrigatórias do domínio alinhamento/sequenciamento genético são exibidas em laranja e o usuário deverá selecionar uma de suas *subfeatures*. No caso do domínio escolhido as *features* inicialmente obrigatórias são o *tipo de aplicação* a ser desenvolvida (*workflow* ou aplicação isolada) e a *plataforma de sequenciamento* a ser utilizada. Conforme exibido na Figura 4.7, a primeira *feature* a ser escolhida foi *workflow*, como tipo de aplicação a ser desenvolvida.

  Após selecionar a *feature workflow* e clicar no botão *Select Feature*, no canto superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo ontológico, ou classe ontológica, correspondente à *feature* que foi selecionada. Neste caso, a *feature workflow* corresponde ao termo/classe de mesmo nome (Figura 4.8).

  Tendo o termo relacionado à *feature* selecionada, as restrições deste termo podem ser buscadas na ontologia. Como exemplo, as restrições associadas aos dois tipos de aplicação possíveis podem ser visualizadas nas Figuras 4.9 (restrições do termo/classe *isolated_application*) e 32 (restrições do termo/classe *workflow*). Ou seja, de acordo com estas restrições, se foi selecionada a *feature workflow*, poderão ser especificadas diversas *bioinformatics_tasks* para compor

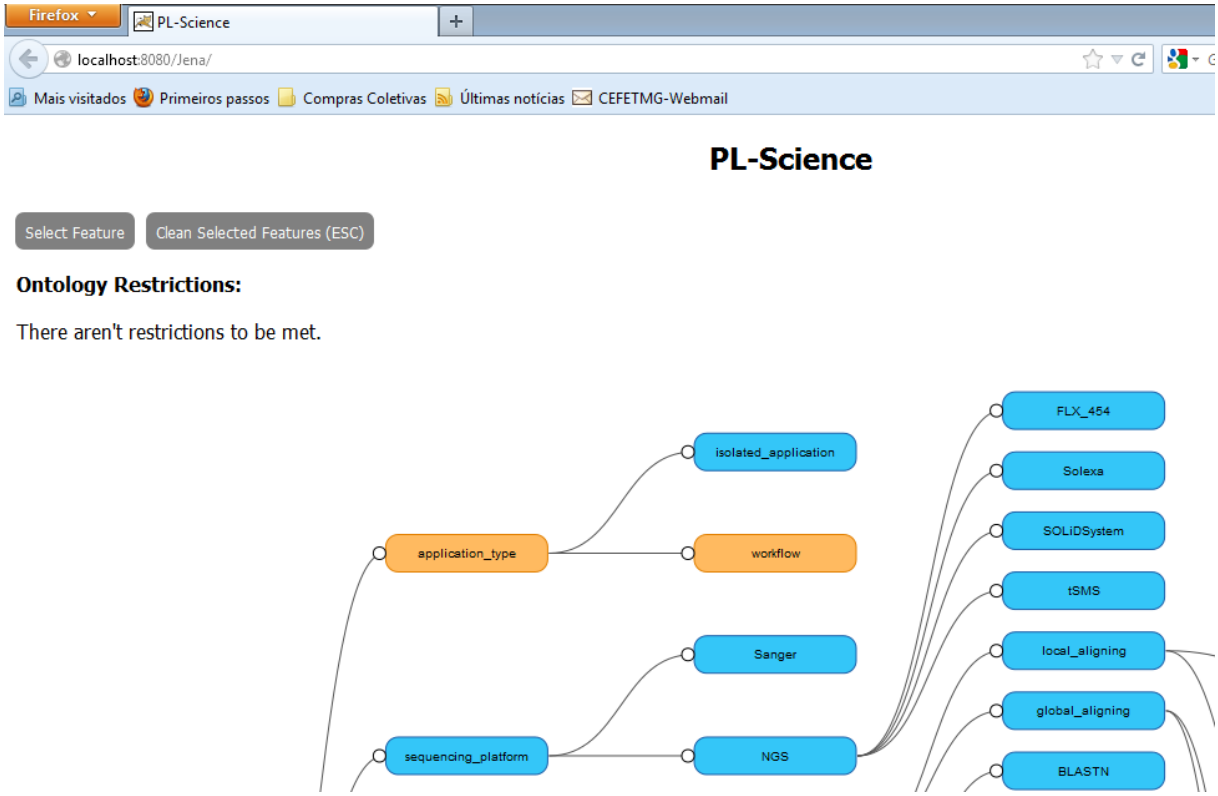

Figura 4.7: Exemplo 1 - Tela 1

```
−<mapping id="50">
    <className>application_type</className>
    <featureName>application_type</featureName>
  </mapping>
−<mapping id="51">
    <className>isolated_application</className>
    <featureName>isolated_application</featureName>
  </mapping>
−<mapping id="52">
    <className>workflow</className>
    <featureName>workflow</featureName>
  </mapping>
```

Figura 4.8: Classe correspondente à *feature workflow*

a aplicação (*workflow*), ao passo que, uma aplicação isolada comporta apenas uma *bioinformatics_task*.

Logo após, é necessária a seleção da Plataforma de Sequenciamento a ser utilizada. O usuário deverá optar pelo sequenciamento pelo método de Sanger, ou pelas novas tecnologias de sequenciamento (NGS).

No caso deste exemplo, escolheu-se a *feature Sanger*, como plataforma de se-

```
<!-- http://gabriellacastro.com.br/SequenceAligningOntology#isolated_application -->

<owl:Class rdf:about="http://gabriellacastro.com.br/SequenceAligningOntology#isolated_application">
    <rdfs:subClassOf rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#application_type"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#workflow_possible_tasks"/>
            <owl:onClass rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#bioinformatics_task"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 4.9: Restrições da Classe *isolated_application*

```
<!-- http://gabriellacastro.com.br/SequenceAligningOntology#workflow -->

<owl:Class rdf:about="http://gabriellacastro.com.br/SequenceAligningOntology#workflow">
    <rdfs:subClassOf rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#application_type"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#workflow_possible_tasks"/>
            <owl:someValuesFrom rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#bioinformatics_task"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 4.10: Restrições da Classe *workflow*

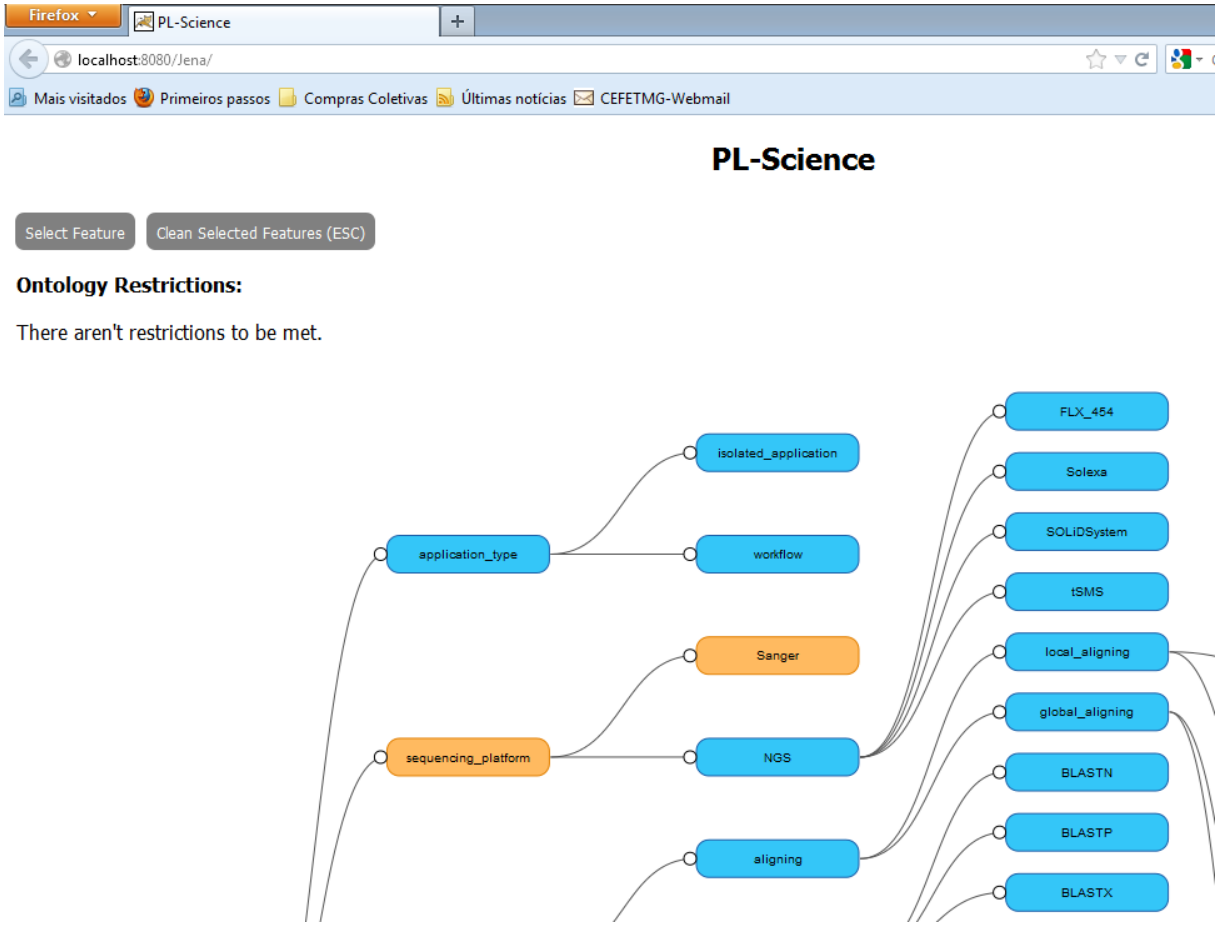

Figura 4.11: Exemplo 1 - Tela 2

quenciamento a ser utilizada (conforme exibido na Figura 4.11). Após selecionar a *feature* e clicar no botão *Select Feature*, no canto superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo/classe correspondente à *feature* que foi selecionada. Neste caso, a *feature Sanger* corresponde à classe de mesmo nome (Figura 4.12).

```
− <mapping id="2">
    <className>bioinformatics_sequencing_platform</className>
    <featureName>sequencing_platform</featureName>
  </mapping>
− <mapping id="3">
    <className>Sanger</className>
    <featureName>Sanger</featureName>
  </mapping>
− <mapping id="4">
    <className>NGS</className>
    <featureName>NGS</featureName>
  </mapping>
```

Figura 4.12: Classe correspondente à *feature Sanger*

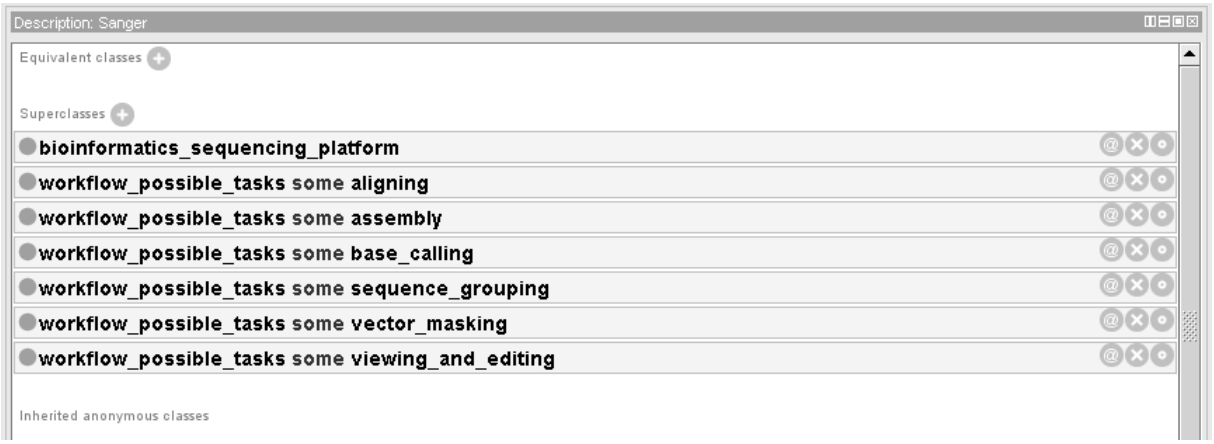

Figura 4.13: Restrições da Classe *Sanger*

Após acessar o arquivo de mapeamento, são analisadas as restrições relacionadas ao termo/classe *Sanger*, que foram definidas na ontologia do domínio, conforme exibido na Figura 4.13. Em seguida, de acordo com as restrições que foram obtidas, é exibida uma nova mensagem ao usuário e a tela de seleção de *features* é alterada (Figura 4.14). A mensagem exibida solicita que o usuário selecione a última tarefa que ele deseja incluir em seu *workflow*. Vale ressaltar que, usando o sequenciamento pelo método de Sanger, todas as tarefas relaci-

onadas no *workflow* base do domínio são possíveis de serem realizadas, o que não ocorre para as plataformas NGS (as tarefas que podem ser realizadas para este tipo de sequenciamento também são expressas através da ontologia).

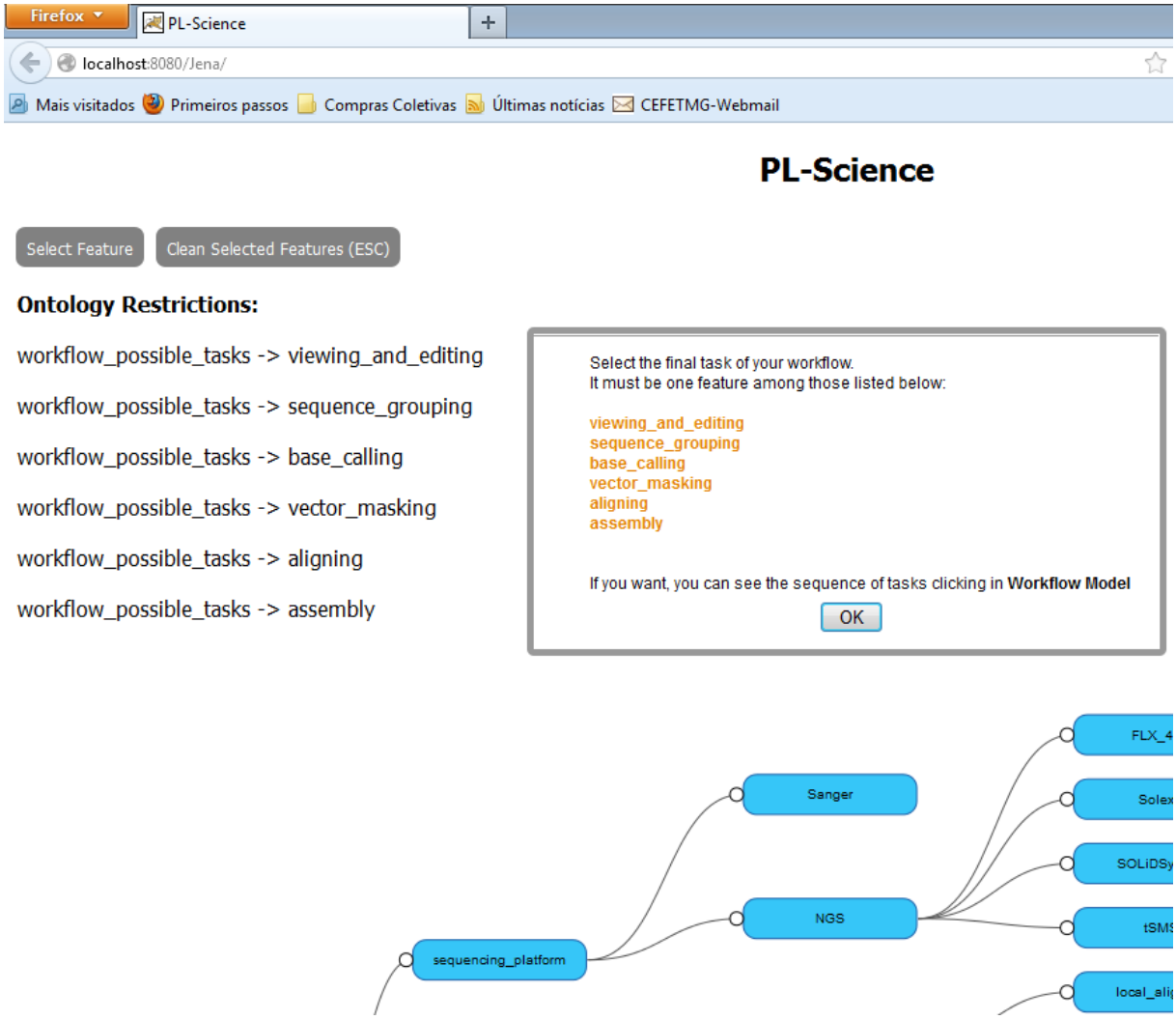

Figura 4.14: Exemplo 1 - Tela 3

Como no caso deste exemplo, tem-se como última tarefa a ser realizada o agrupamento de sequências (*sequence_grouping*), esta *feature* foi selecionada, conforme exibido na Figura 4.15.

– **Passo 2**: Para cada tarefa do *workflow*, são analisadas as possibilidades de variação, e, em seguida, o usuário define o algoritmo, o *Web Service* ou a aplicação que será 'instanciado' no *workflow* base.

Na abordagem PL-Science, todos os algoritmos, *Web Services* ou outras aplicações que poderão vir a ser utilizados para compor algum *workflow* a ser gerado a partir do Modelo Abstrato de *Workflow* devem ser catalogados no repositório

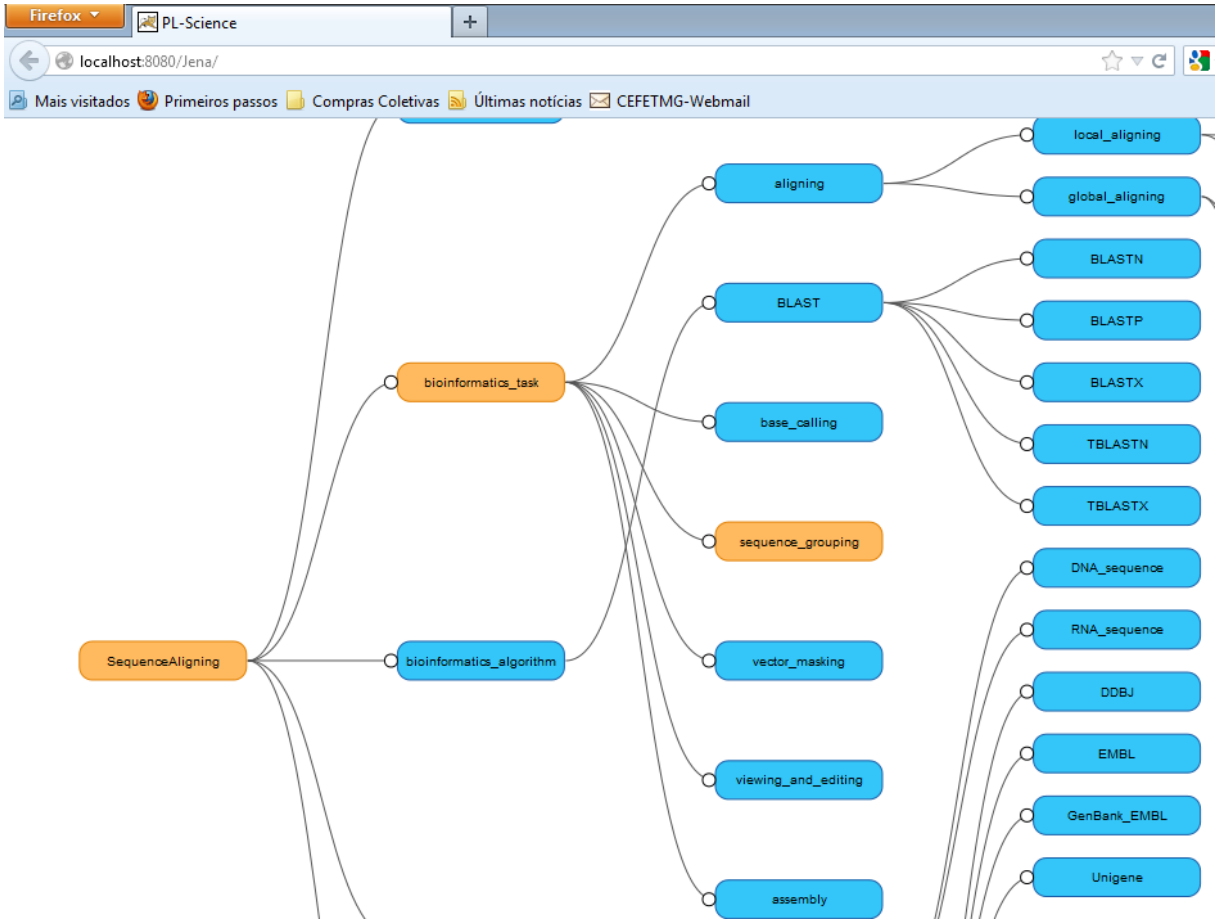

Figura 4.15: Exemplo 1 - Seleção da última tarefa do *workflow* a ser criado

do domínio. A gerência deste repositório é feita a partir da ontologia de domínio. Assim, os artefatos catalogados são registrados na ontologia de domínio como *indivíduos* desta. Atualmente, nesta primeira versão da PL-Science basta inserir o algoritmo, *Web Service* ou aplicação como um indivíduo, dando um nome a este, especificar o seu tipo (se é um algoritmo, *Web Service* ou outro tipo de aplicação) e sua localização (se for um *Web Service*, deve-se especificar onde este pode ser encontrado ou, se for um determinado algoritmo ou aplicação, qual a localização deste no repositório de artefatos, usando uma url, por exemplo). Além disto, deve ser especificada para cada um dos indivíduos uma das subclasses de *bioinformatics_algorithm*, já que, na ontologia, são estas as subclasses que se relacionam com as *bioinformatics_tasks* que fazem parte do modelo abstrato de *workflow*. A Figura 4.16 apresenta um *Web Service* que foi cadastrado como indivíduo na ontologia, a sua localização no repositório Biocatalogue (BHAGAT et al., 2010) e a associação deste indivíduo à classe

PHRAP, que corresponde à subclasse de *bioinformatics_algorithm* da qual ele faz parte. Vale ressaltar que diversos outros algoritmos/ *Web Services* ou aplicações podem ser associados a esta mesma classe *PHRAP*, de acordo com a função que realizam.

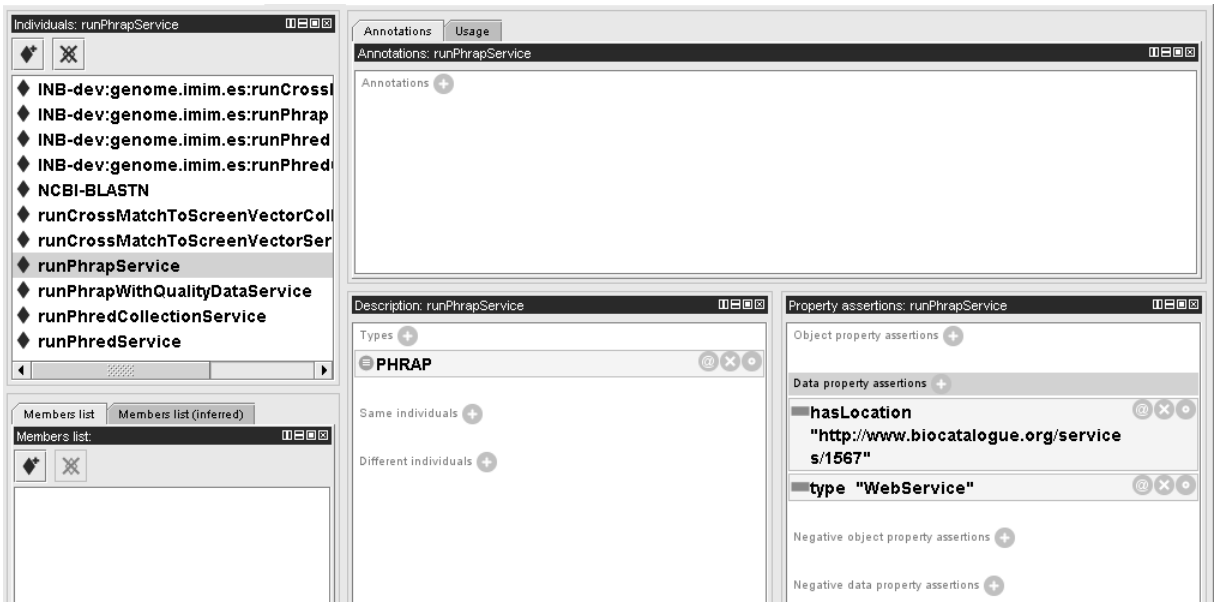

Figura 4.16: Indivíduo *runPhrapService*

Continuando no **Passo 2** desta fase, a aplicação realiza diversas buscas na ontologia, analisando cada uma das restrições de cada uma das tarefas do modelo de *workflow* que foram escolhidas. De acordo com este exemplo, a primeira tarefa, *base_calling* será analisada, e, em seguida, as tarefas *vector_masking* e *sequence_grouping*.

Estas buscas na ontologia funcionam da seguinte maneira: em um primeiro momento, busca-se por uma subclasse de *bioinformatics_algorithm* que esteja relacionada ao *base_calling* (já que esta foi a primeira tarefa selecionada no modelo de *workflow*) por meio da *ObjectProperty performs_task*. Após encontrada ao menos uma subclasse nesta busca, usa-se uma máquina de inferência na ontologia de domínio de forma a verificar se existe alguma outra subclasse de *bioinformatics_algorithm* que seja 'igual' à primeira subclasse que foi encontrada na busca inicial (é justamente neste ponto de aplicação da abordagem que encontra-se a grande vantagem da utilização do mecanismo de inferência proporcionado pelas ontologias, ou seja, é ele que auxilia na apresentação de todas as 'possibilidades de variação' de 'algoritmos' que poderão realizar a tarefa

selecionada no Modelo Abstrato de *Workflow*). De acordo com as Figuras 4.17 e 4.18, após executar a máquina de inferência, não foi encontrada nenhuma outra classe que fosse 'igual' à primeira classe encontrada na busca inicial, porém, a Figura 4.19, já mostra que as classes *PHRAP* e *CAP3* são 'iguais'. Para a abordagem PL-Science, isto significa que, tanto os indivíduos catalogados para a classe *CAP3* como os catalogados para a classe *PHRAP* poderão ser exibidos ao usuário como uma possibilidade de algoritmo/*Web Service*/aplicação que será capaz de executar a tarefa de agrupamento de sequências (*sequence_grouping*).

Figura 4.17: Exemplo de Inferência sobre a Classe *PHRED*

Figura 4.18: Exemplo de Inferência sobre a Classe *Cross_match*

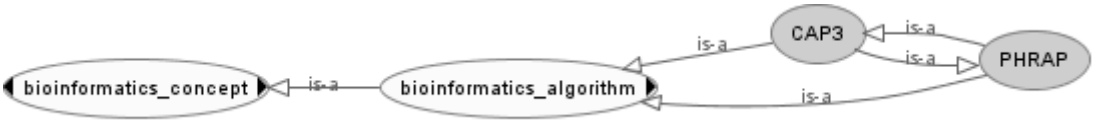

Figura 4.19: Exemplo de Inferência sobre a Classe *PHRAP*

Após a análise e as buscas realizadas na ontologia, conforme apresentado anteriormente, são mostrados ao usuário (em forma de uma listagem), todos os algoritmos, *Web Services* ou aplicações específicas que foram encontrados para cada uma das tarefas selecionadas para compor o *workflow*. Neste caso, as tarefas T1, T2 e T3. Neste passo, o usuário poderá definir uma das possibilidades listadas (algoritmos/*Web Services* ou aplicações) para cada uma das tarefas.

– **Passo 3**: Geração de um arquivo especificado em XML, com o detalhamento das tarefas que deverão compor o *workflow* científico.

Após a definição do algoritmo, *Web Service* ou aplicação definido pelo usuário no **Passo 2** para cada uma das tarefas do *workflow*, como um último passo, a abordagem PL-Science gera um arquivo especificado em XML, que apresenta

o encadeamento dos algoritmos, *Web Service* ou aplicações de acordo com as tarefas que deverão ser realizadas. No caso deste exemplo, a Figura 4.20 apresenta o arquivo gerado pela aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow>
    <Processes>
        <Process name="INB-dev:genome.imim.es:runPhred" id="Process_1">
            <Parameters>
                <Parameter name="type" value="WebService" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/2290"/>
            </Parameters>
        </Process>
        <Process name="runCrossMatchToScreenVectorService" id="Process_2">
            <Parameters>
                <Parameter name="type" value="WebService" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/1586<"/>
            </Parameters>
        </Process>
        <Process name="INB-dev:genome.imim.es:runPhrap" id="Process_3">
            <Parameters>
                <Parameter name="type" value="WebService" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/2268"/>
            </Parameters>
        </Process>
    </Processes>
    <Connections>
        <Connection name="Connection_1" source="Process_1" target=" Process_2"/>
        <Connection name="Connection_2" source="Process_2" target="Process_3"/>
    </Connections>
</Workflow>
```

Figura 4.20: Arquivo XML gerado pela Aplicação (Produto 1)

4.3.2 PRODUTO 2 - TAREFA ISOLADA PARA ALINHAMENTO MÚLTIPLO DE SEQUÊNCIAS

Nesta segunda especificação de um produto a partir da LPSC no domínio de sequenciamento/alinhamento genético, tem-se como objetivo a criação de uma aplicação específica para alinhamento global de múltiplas sequências.

O Alinhamento de Múltiplas Sequências (ou *Multiple Sequence Alignment - MSA*) consiste no alinhamento de três ou mais sequências biológicas (sendo estas sequências de proteína ou sequências nucleotídicas) de comprimentos similares. A utilização de programas que realizam este tipo de alinhamento possibilita a identificação de regiões de similaridade entre as sequências que podem indicar relações funcionais, estruturais e/ou evolutivas entre as sequências estudadas (EMBL-EBI, 2012).

Este produto envolve apenas a tarefa de Alinhamento (T5) do Modelo Abstrato de *Workflow* do domínio, conforme exibido na Figura 4.21.

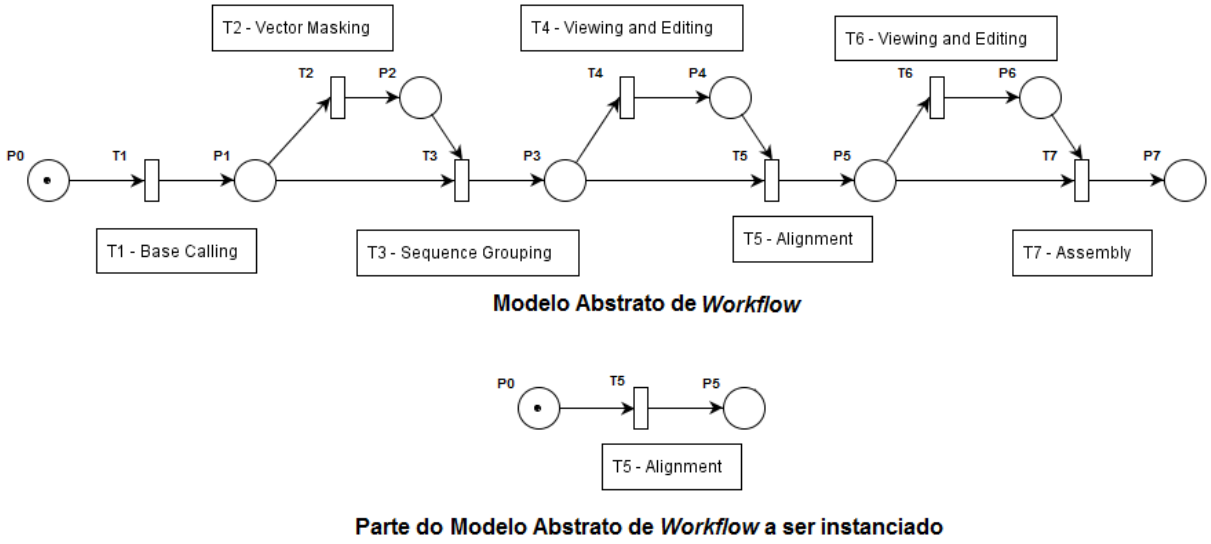

Figura 4.21: Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 2)

Através da metodologia da abordagem, os passos seguidos são detalhados a seguir:

– **Passo 1**: As *features* inicialmente obrigatórias do domínio alinhamento/sequenciamento genético são exibidas em laranja (Figura 4.22) e o usuário deverá selecionar uma de suas *subfeatures*. No caso do domínio escolhido, as *features* inicialmente obrigatórias são o *tipo de aplicação* a ser desenvolvida (*workflow* ou aplicação isolada) e a *plataforma de sequenciamento* a ser utilizada. Conforme exibido na Figura 4.22, a primeira *feature* a ser escolhida foi *isolated_application*, como tipo de aplicação a ser desenvolvida, já que se tem como objetivo a realização de apenas uma tarefa, o alinhamento.

  Após selecionar a *feature workflow* e clicar no botão *Select Feature*, no canto superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo ontológico ou classe ontológica correspondente à *feature* que foi selecionada. Neste caso, a *feature isolated_application* corresponde ao termo/classe de mesmo nome (Figura 4.23).

  Tendo o termo relacionado à *feature* selecionada, as restrições deste termo podem ser buscadas na ontologia. Conforme exibido na Figura 4.24, as restrições associadas ao termo/classe *isolated_application* especifica que uma aplicação isolada comporta apenas uma *bioinformatics_task*.

  Logo após, é necessária a seleção da Plataforma de Sequenciamento a ser utilizada (já que esta é uma *feature* inicialmente obrigatória no modelo de *features*

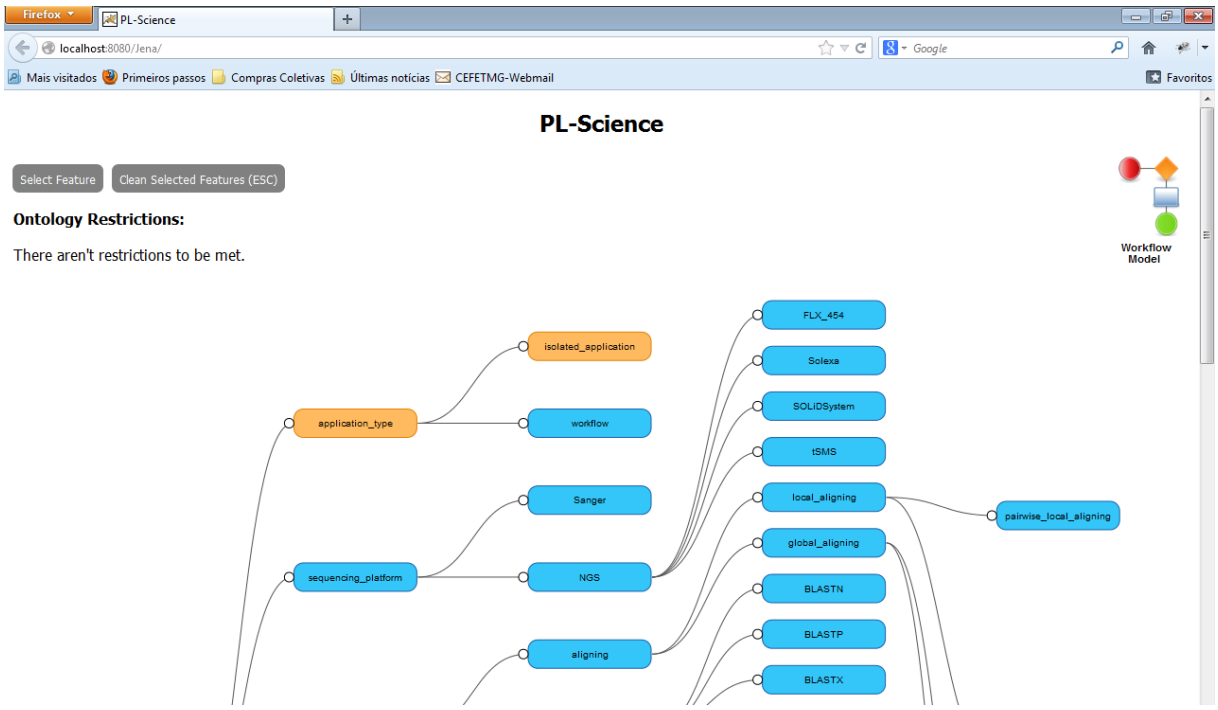

Figura 4.22: Exemplo 2 - Tela 1

```
−<mapping id="50">
    <className>application_type</className>
    <featureName>application_type</featureName>
  </mapping>
−<mapping id="51">
    <className>isolated_application</className>
    <featureName>isolated_application</featureName>
  </mapping>
−<mapping id="52">
    <className>workflow</className>
    <featureName>workflow</featureName>
  </mapping>
```

Figura 4.23: Classe correspondente à *feature isolated_application*

```
<!-- http://gabriellacastro.com.br/SequenceAligningOntology#isolated_application -->

<owl:Class rdf:about="http://gabriellacastro.com.br/SequenceAligningOntology#isolated_application">
    <rdfs:subClassOf rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#application_type"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#workflow_possible_tasks"/>
            <owl:onClass rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#bioinformatics_task"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 4.24: Restrições da Classe *isolated_application*

do domínio em questão). No momento da criação deste segundo produto, optou-se pelo sequenciamento de Nova Geração (NGS), portanto, foi selecionada a *feature* NGS, conforme exibido na Figura 4.25. Após selecionar a *feature* e clicar no botão *Select Feature*, no canto superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo/-classe correspondente à *feature* que foi selecionada. Neste caso, a *feature* NGS corresponde à classe de mesmo nome (Figura 48).

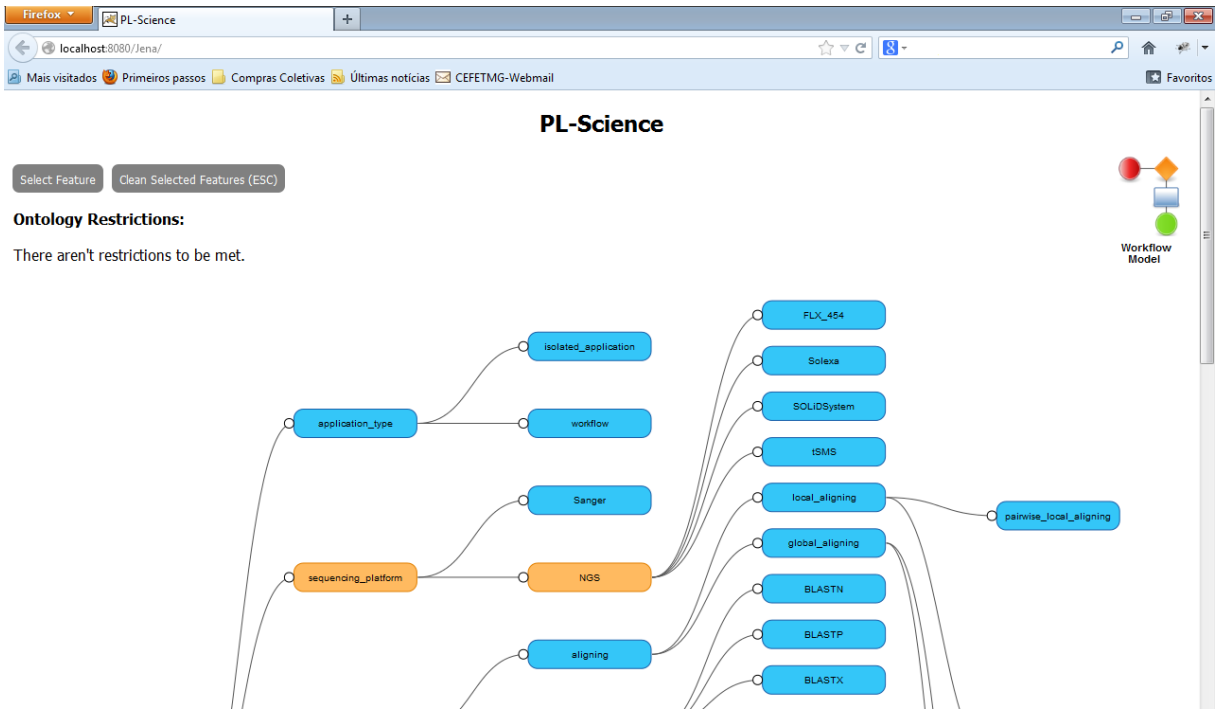

Figura 4.25: Exemplo 2 - Tela 2

```
- <mapping id="2">
    <className>bioinformatics_sequencing_platform</className>
    <featureName>sequencing_platform</featureName>
  </mapping>
- <mapping id="3">
    <className>Sanger</className>
    <featureName>Sanger</featureName>
  </mapping>
- <mapping id="4">
    <className>NGS</className>
    <featureName>NGS</featureName>
  </mapping>
```

Figura 4.26: Classe correspondente à *feature NGS*

Após acessar arquivo de mapeamento, são analisadas as restrições relacionadas o termo/classe NGS, que foram definidas na ontologia do domínio, conforme exibido na Figura 4.27. Em seguida, de acordo com as restrições que foram obtidas, é exibida uma nova mensagem ao usuário e a tela de seleção de *features* é alterada (Figura 4.28). A mensagem exibida solicita que o usuário selecione a tarefa que irá compor sua aplicação isolada.

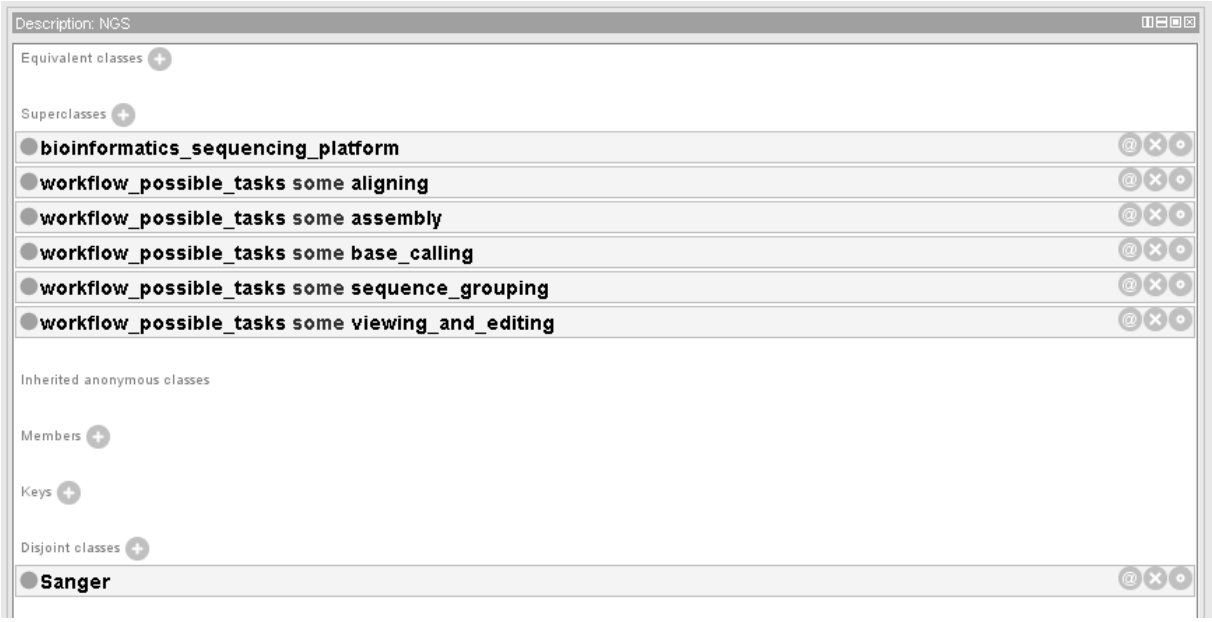

Figura 4.27: Restrições da Classe *NGS*

Como neste exemplo tem-se como objetivo a realização da tarefa de alinhamento global múltiplo, esta *feature* foi selecionada, conforme exibido na Figura 4.29.

– **Passo 2**: Para cada tarefa do *workflow*, são analisadas as possibilidades de variação e o usuário define qual o algoritmo, *Web Service* ou aplicação que será 'instanciado' no *workflow* base.

Neste passo, como não se trata de um *workflow* a ser gerado e sim de uma aplicação isolada, a aplicação realizará buscas na ontologia, analisando apenas a única tarefa do Modelo Abstrato de *Workflow* que foi escolhida. De acordo com este exemplo, a tarefa *multiple_global_aligning* será analisada. Caso o usuário não selecione *multiple_global_aligning* (*subfeature* de *aligning*) na tela exibida na Figura 4.29 e selecione apenas a tarefa *aligning*, a aplicação solicitará ao mesmo que especifique melhor o tipo de alinhamento desejado, ou seja,

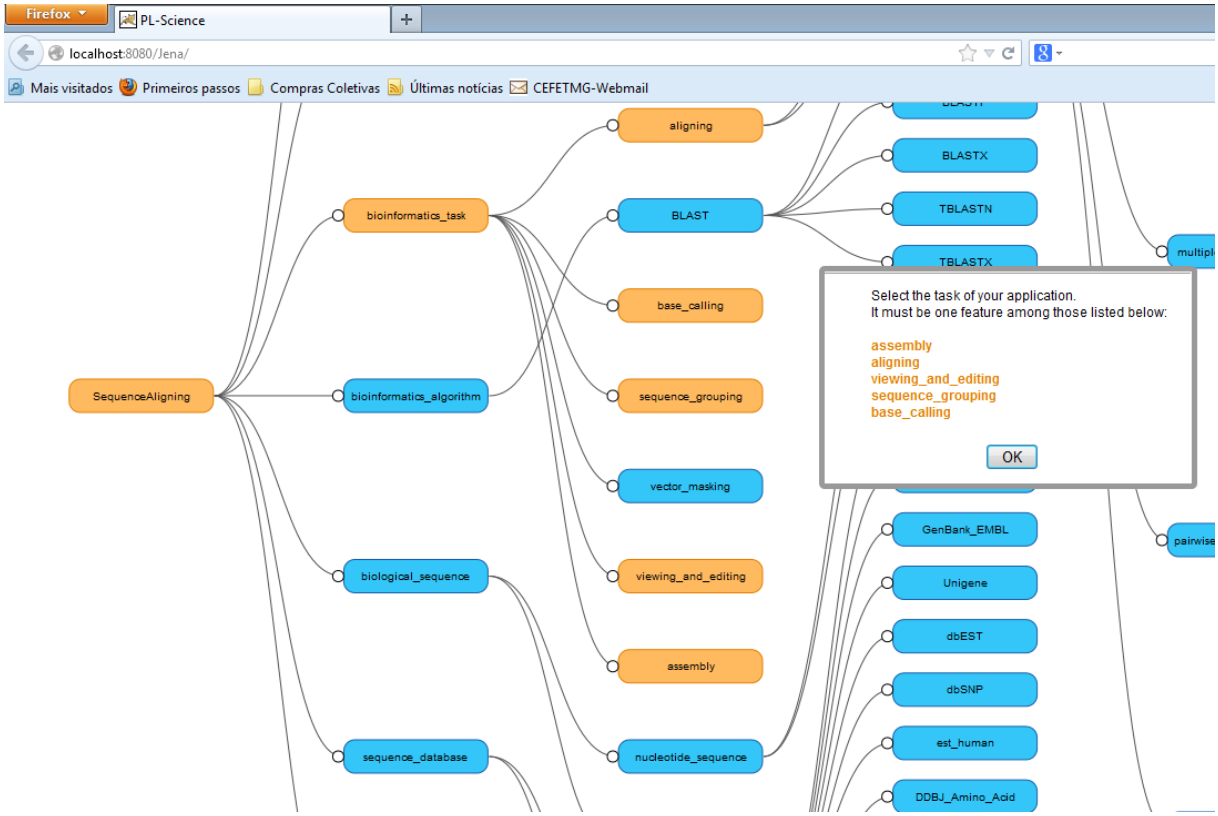

Figura 4.28: Exemplo 2 - Tela 3

selecione uma das *subfeatures* de *aligning*.

No caso deste exemplo, a busca na ontologia se dará da seguinte forma (mesmo processo do Produto 1, só que apenas para uma tarefa específica): em um primeiro momento, busca-se por uma subclasse de *bioinformatics_algorithm* que esteja relacionada ao alinhamento global múltiplo (*multiple_global_aligning*) por meio da *ObjectProperty performs_task*. Depois de encontrada ao menos uma subclasse nesta busca, utiliza-se uma máquina de inferência na ontologia de domínio de forma a verificar se existe alguma outra subclasse de *bioinformatics_algorithm* que seja 'igual' à primeira subclasse que foi encontrada na busca inicial. Neste exemplo, encontrou-se inicialmente a classe *ClustalW2*. Logo após, de acordo com a Figura 4.30, depois de executar a máquina de inferência, tem-se que a classe ClustalW2 passa a ser uma subclasse de Clustal_Omega. Portanto, poderão ser exibidos ao usuário tanto os indivíduos catalogados para a classe *ClustalW2* como os catalogados para a classe *Clustal_Omega* como uma possibilidade de algoritmo/*Web Service*/aplicação que será capaz de executar a tarefa de alinhamento global múltiplo de sequências (*multiple_global_aligning*).

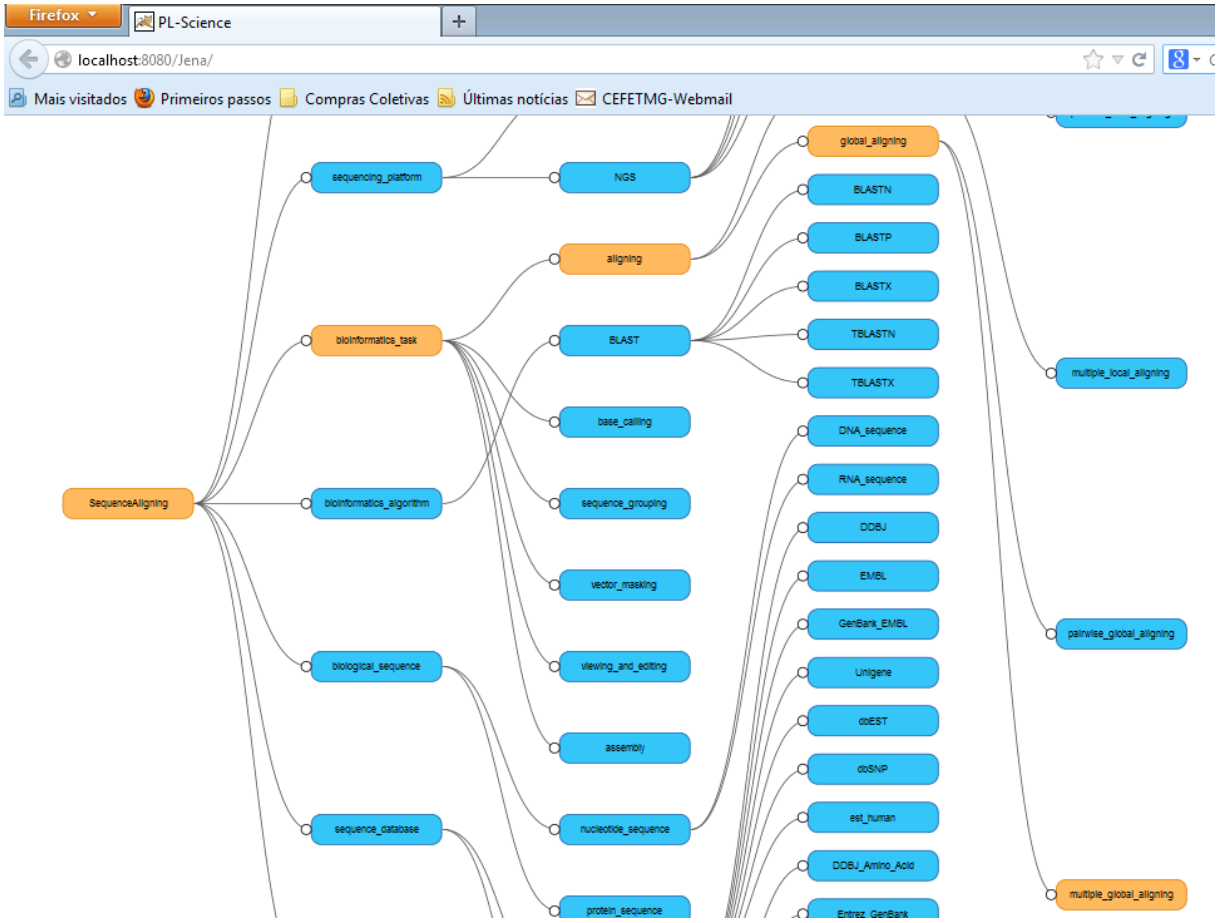

Figura 4.29: Exemplo 2 - Seleção da tarefa desejada

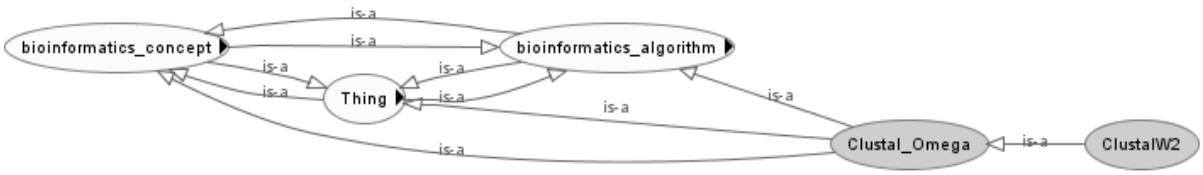

Figura 4.30: Exemplo de Inferência sobre a Classe *ClustalW2*

Neste exemplo, vale ressaltar que, como as classes não se apresentaram como equivalentes (iguais) após a execução do mecanismo de inferência, a diferença entre estas é apresentada ao usuário, de forma que ele possa estabelecer corretamente o algoritmo/*Web Service* ou aplicação a ser utilizada.

As Figuras 4.31 e 4.32 apresentam a restrição que diferencia ambas as classes (tipos de sequências de entrada suportadas), *Clustal_Omega* e *ClustalW2*, respectivamente:

Portanto, após a análise e as buscas realizadas na ontologia, conforme apresentado anteriormente, será mostrado ao usuário todos os algoritmos, *Web Services*

Figura 4.31: Sequências de entrada suportadas pelo *Clustal_Omega*

Figura 4.32: Sequências de entrada suportadas pelo *ClustalW2*

ou aplicações específicas que foram encontrados para a tarefa de alinhamento (T5), mais especificamente para o alinhamento global múltiplo, sejam estes relacionados à classe *ClustalW2* ou *Clustal_Omega*. Neste passo, o usuário poderá definir uma das possibilidades listadas (algoritmos/ *Web Services* ou aplicações) para compor a sua aplicação específica. No caso deste Produto, optou-se por escolher uma aplicação Web que realiza a tarefa de alinhamento global múltiplo para sequência de DNA, disponibilizada pelo Instituto Europeu de Bioinformática (*European Bioinformatics Institute* - EBI) (EMBL-EBI, 2012), conforme pode ser visualizado na Figura 4.33.

– **Passo 3**: Geração de um arquivo especificado em XML, com o detalhamento das tarefas que deverão compor o *workflow* científico.

  Após a definição do algoritmo, *Web Service* ou aplicação definido pelo usuário no **Passo 2** para a tarefa a ser realizada como aplicação isolada, como um último passo, a abordagem PL-Science gera um arquivo especificado em XML, que apresenta as principais propriedades do Algoritmo/WebService ou aplicação a ser usada. No caso deste exemplo, a Figura 4.33 apresenta o arquivo gerado pela aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow>
    <Processes>
        <Process name="ClustalW2_-_Multiple_Sequence_Alignment" id="Process_1">
            <Parameters>
                <Parameter name="type" value="Application"/>
                <Parameter name="hasLocation" value="http://www.ebi.ac.uk/Tools/msa/clustalw2/"/>
            </Parameters>
    </Processes>
    <Connections>
    </Connections>
</Workflow>
```

Figura 4.33: Arquivo XML gerado pela Aplicação (Produto 2)

4.3.3 PRODUTO 3 - *WORKFLOW* PARA SEQUENCIAMENTO E ALINHAMENTO

Como terceiro produto a ser gerado a partir da LPSC proposta, tem-se como objetivo a criação de um *workflow* que trata desde a obtenção dos dados de uma sequência, através de um sequenciador, até a comparação desta sequência obtida com bases de sequências já existentes, através da tarefa de alinhamento. Ele inclui parte do *workflow* que foi gerado no Produto 1, porém, agrega a tarefa de alinhamento, que realiza a comparação da sequência gerada com sequências já existentes. O Produto 3 é apresentado como um dos exemplos da abordagem já que o Produto 1 não realiza nenhuma tarefa de comparação (ou alinhamento) entre a sequência gerada no processo de sequenciamento e outras sequências existentes. Este produto gerado a partir da LPSC envolve desde a tarefa de *Base Calling* (T1) até a tarefa de Alinhamento (T5) do Modelo Abstrato de *Workflow* do domínio, conforme exibido na Figura 4.34. Portanto, o principal diferencial entre este produto e o Produto 1 é a inclusão da tarefa T5, responsável pelo alinhamento entre as sequências.

Através da metodologia da abordagem, os passos seguidos são detalhados a seguir:

- **Passo 1**: As *features* inicialmente obrigatórias do domínio alinhamento/sequenciamento genético são exibidas em laranja e o usuário deverá selecionar uma de suas *subfeatures*. No caso do domínio escolhido, as *features* inicialmente obrigatórias são o *tipo de aplicação* a ser desenvolvida (*workflow* ou aplicação isolada) e a *plataforma de sequenciamento* a ser utilizada. Conforme exibido na Figura 4.35, a primeira *feature* a ser escolhida foi *workflow*, como tipo de aplicação a ser desenvolvida.

  Após selecionar a *feature workflow* e clicar no botão *Select Feature*, no canto

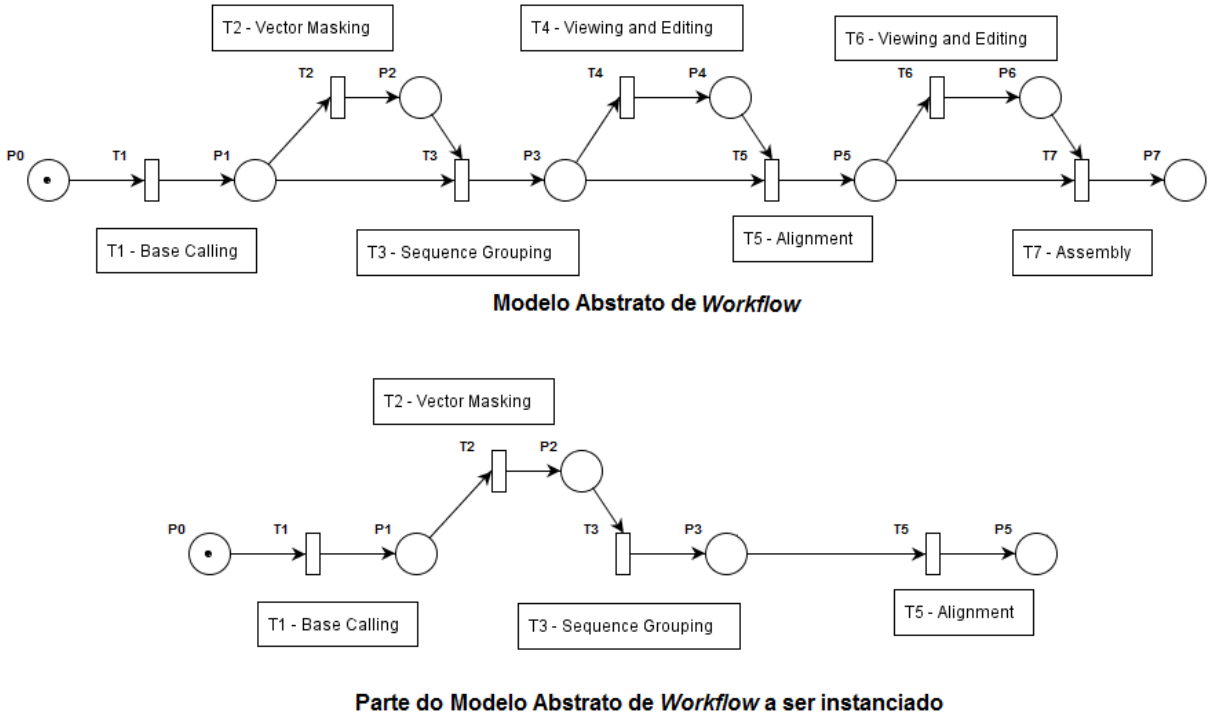

Figura 4.34: Parte do Modelo Abstrato de *Workflow* a ser instanciado (Produto 3)

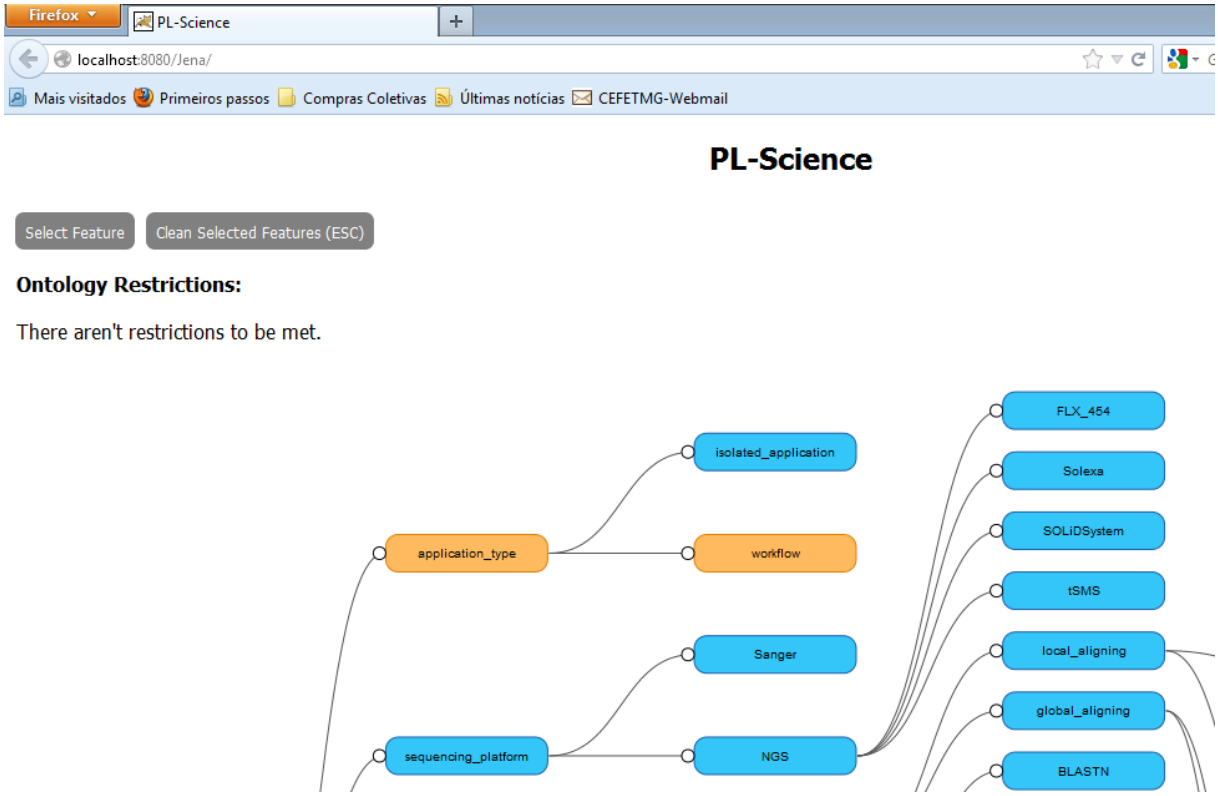

Figura 4.35: Exemplo 3 - Tela 1

superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo ontológico ou classe ontológica correspondente à

feature que foi selecionada. Neste caso, a *feature workflow* corresponde ao termo/classe de mesmo nome (Figura 4.36).

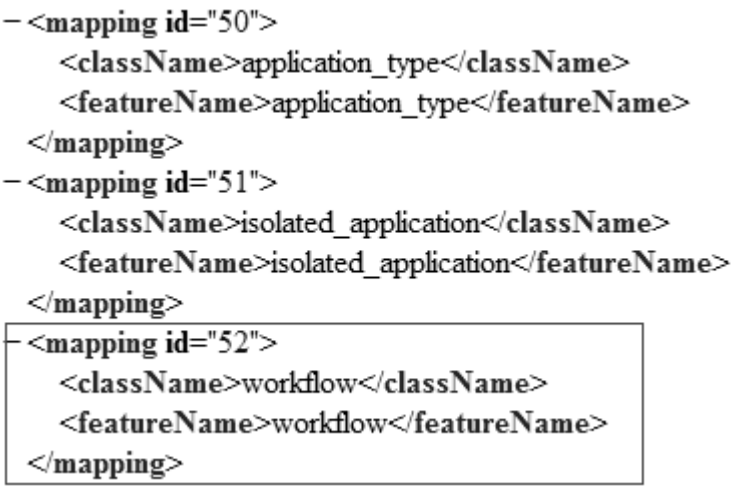

Figura 4.36: Classe correspondente à *feature workflow*

Tendo o termo relacionado à *feature* selecionada, as restrições relacionadas a este termo podem ser buscadas na ontologia. Como exemplo, de acordo com as restrições associadas a *feature workflow* (Figura 4.37), poderão ser especificadas diversas *bioinformatics_tasks* para compor a aplicação (*workflow*).

```
<!-- http://gabriellacastro.com.br/SequenceAligningOntology#workflow -->

<owl:Class rdf:about="http://gabriellacastro.com.br/SequenceAligningOntology#workflow">
    <rdfs:subClassOf rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#application_type"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#workflow_possible_tasks"/>
            <owl:someValuesFrom rdf:resource="http://gabriellacastro.com.br/SequenceAligningOntology#bioinformatics_task"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 4.37: Restrições da Classe *workflow*

Logo após, o usuário deverá selecionar o método de sequenciamento desejado. No caso deste exemplo, escolheu-se a *feature Sanger*, como plataforma de sequenciamento a ser utilizada (conforme exibido na Figura 4.38). Após selecionar a *feature* e clicar no botão *Select Feature*, no canto superior esquerdo da tela da aplicação, o arquivo de mapeamento é analisado, de forma a obter o termo/classe correspondente à *feature* que foi selecionada. Neste caso, a *feature Sanger* corresponde à classe de mesmo nome (Figura 4.39).

Após acessar arquivo de mapeamento, são analisadas as restrições relacionadas o termo/classe *Sanger*, que foram definidas na ontologia do domínio, conforme

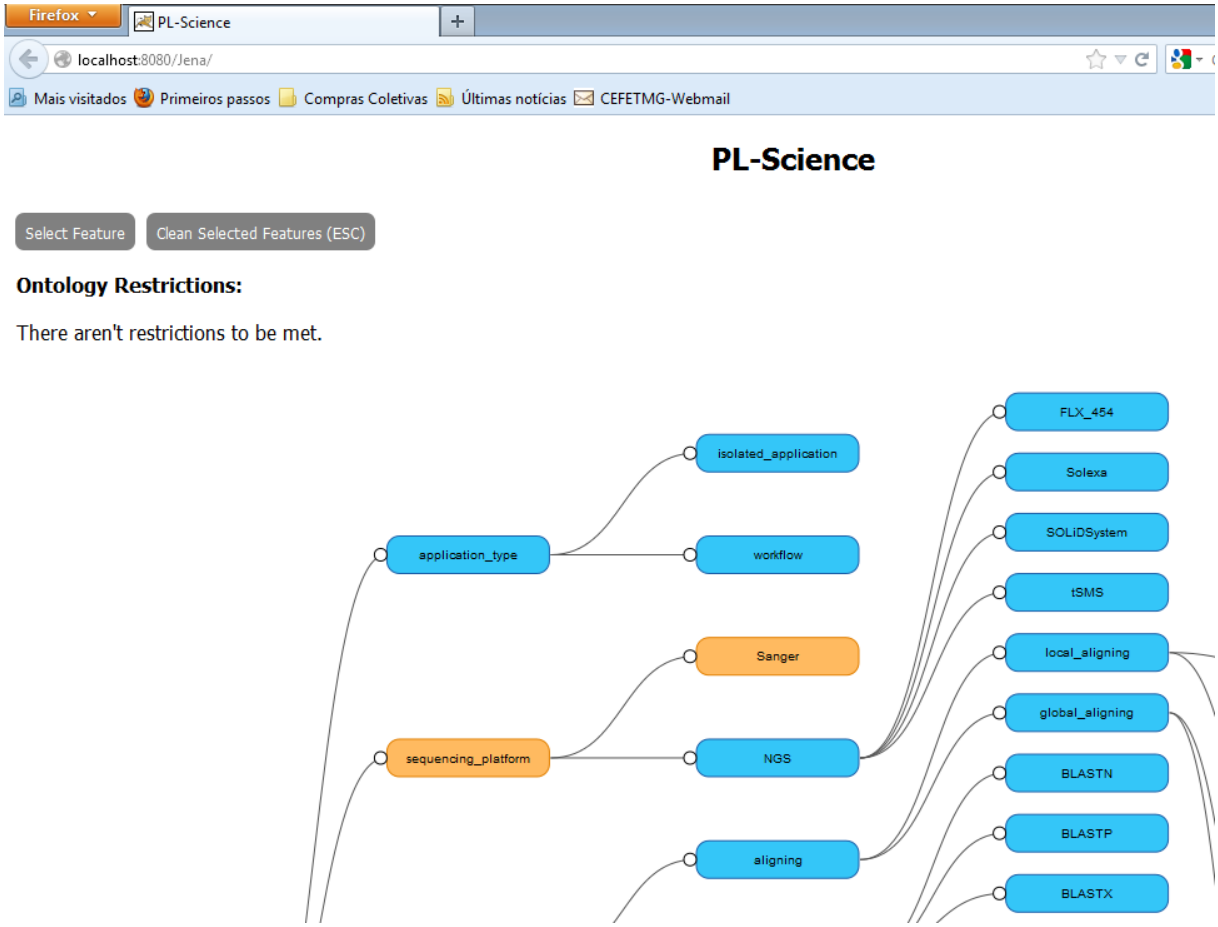

Figura 4.38: Exemplo 3 - Tela 2

```
−<mapping id="2">
    <className>bioinformatics_sequencing_platform</className>
    <featureName>sequencing_platform</featureName>
  </mapping>
−<mapping id="3">
    <className>Sanger</className>
    <featureName>Sanger</featureName>
  </mapping>
−<mapping id="4">
    <className>NGS</className>
    <featureName>NGS</featureName>
  </mapping>
```

Figura 4.39: Classe correspondente à *feature Sanger*

exibido na Figura 4.40. Em seguida, de acordo com as restrições que foram obtidas, é exibida uma nova mensagem ao usuário e a tela de seleção de *features* é alterada (Figura 4.41). A mensagem solicita que o usuário selecione a última tarefa que ele deseja incluir em seu *workflow*.

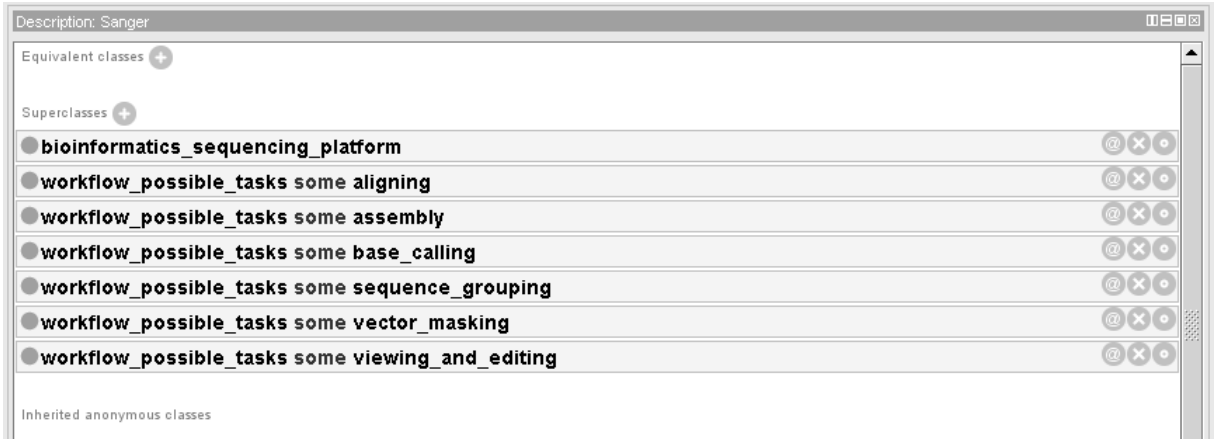

Figura 4.40: Restrições da Classe *Sanger*

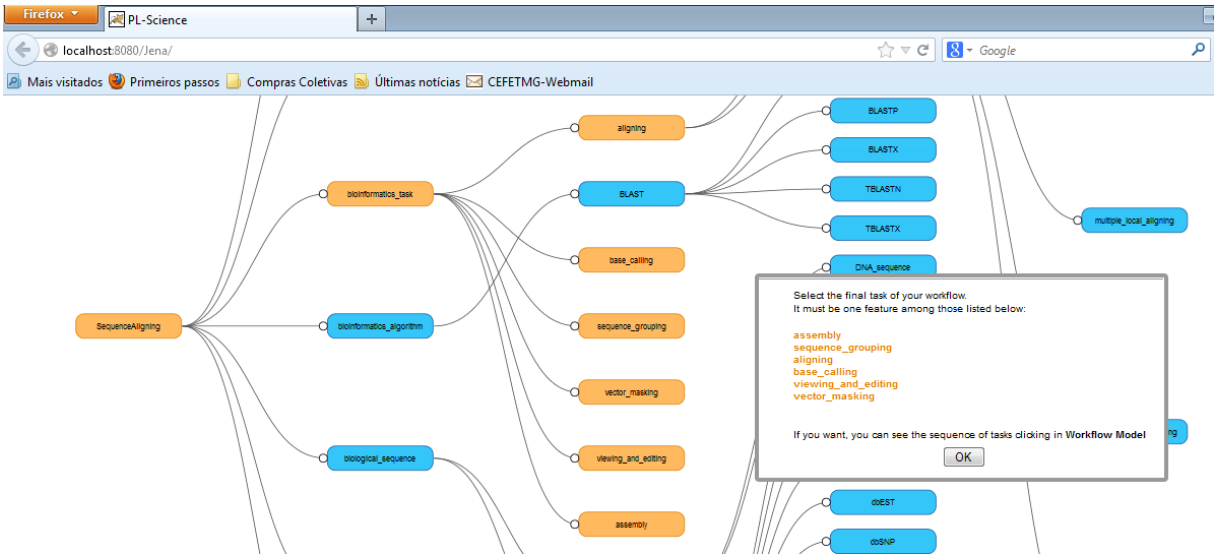

Figura 4.41: Exemplo 3 - Tela 3

Como no caso deste exemplo, tem-se como última tarefa a ser realizada o alinhamento (*aligning*), esta *feature* foi selecionada, conforme é exibido na Figura 4.42.

– **Passo 2**: Para cada tarefa do *workflow*, são analisadas as possibilidades de variação e o usuário define qual o algoritmo, *Web Service* ou aplicação que será 'instanciado' no *workflow* base.

Neste passo, a aplicação realiza diversas buscas na ontologia, analisando cada uma das restrições de cada uma das tarefas do modelo de *workflow* que foram escolhidas. De acordo com este exemplo, a primeira tarefa, *base_calling* será analisada, e, em seguida, as tarefas *vector_masking* , *sequence_grouping* e *aligning*.

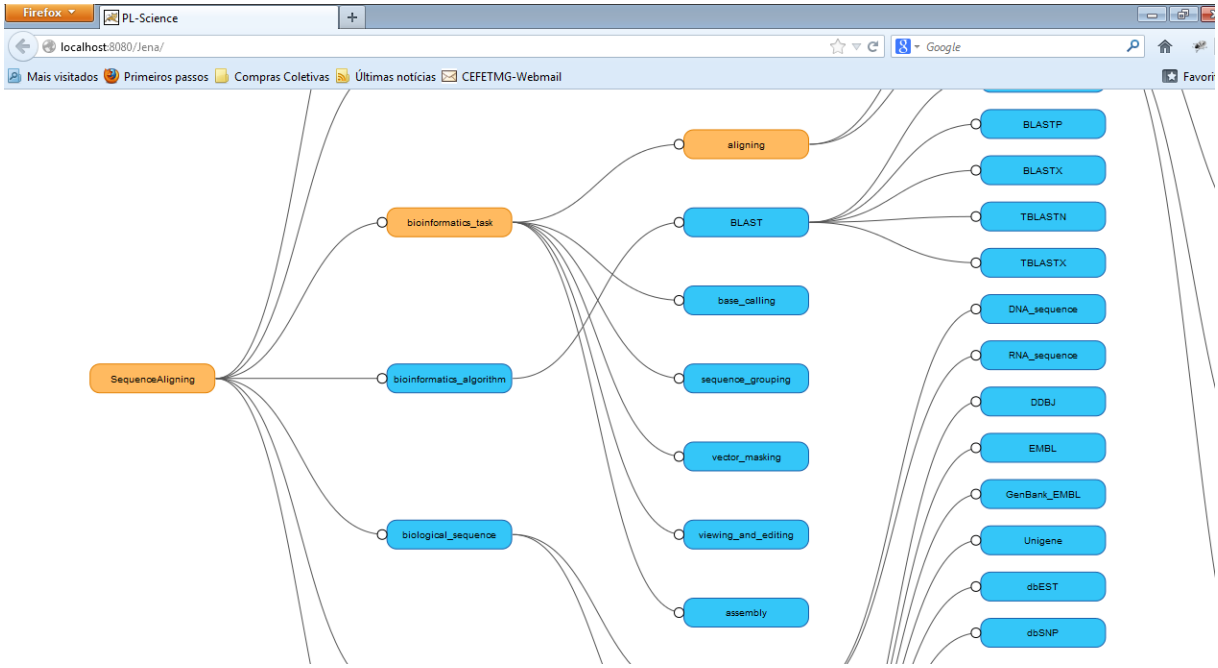

Figura 4.42: Seleção da *feature* que representa a última tarefa do *workflow*

Estas buscas na ontologia funcionam da seguinte maneira: em um primeiro momento, busca-se por uma subclasse de *bioinformatics_algorithm* que esteja relacionada ao *base_calling* (já que esta foi a primeira tarefa selecionada no modelo de *workflow*) por meio da *ObjectProperty performs_task*. Após encontrada ao menos uma subclasse nesta busca, usa-se uma máquina de inferência na ontologia de domínio, de forma a encontrar se existe alguma outra subclasse de *bioinformatics_algorithm* que seja 'igual' à primeira subclasse que foi encontrada na busca inicial. De acordo com as Figuras 4.43 e 4.44, após executar a máquina de inferência, não foi encontrada nenhuma outra classe que fosse 'igual' à primeira classe encontrada na busca inicial, porém, a Figura 4.45 mostra que as classes *PHRAP* e *CAP3* são 'iguais'. Para a abordagem PL-Science, isto significa que tanto os indivíduos catalogados para a classe *CAP3* como os catalogados para a classe *PHRAP* poderão ser exibidos ao usuário como uma possibilidade de algoritmo/*Web Service*/aplicação, que será capaz de executar a tarefa de agrupamento de sequências (*sequence_grouping*).

Figura 4.43: Exemplo de Inferência sobre a Classe *PHRED*

Figura 4.44: Exemplo de Inferência sobre a Classe *Cross_match*

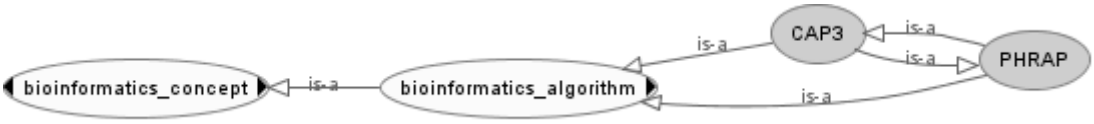

Figura 4.45: Exemplo de Inferência sobre a Classe *PHRAP*

Já para a realização da tarefa de alinhamento, não foi encontrada nenhuma subclasse de *bioinformatics_algorithm* relacionada à classe *aligning*. Assim, ao deparar-se com esta situação, a aplicação desenvolvida verifica se a classe *aligning* possui alguma subclasse definida. Caso positivo (como é o caso deste exemplo), é solicitado ao usuário que especifique melhor o tipo de alinhamento desejado. Ou seja, ele deverá selecionar uma das *subfeatures* de aligning, conforme exibido na Figura 4.46.

No caso deste exemplo, optou-se pelo alinhamento local duplo (*pairwise_local_aligning*) e esta *feature* foi selecionada. Já para esta tarefa, foram encontradas subclasses de *bioinformatics_algorithm* relacionadas. Após executar o mecanismo de inferência, obteve-se, conforme Figura 4.47, que as classes *BLAST* e *SmithWaterman* são equivalentes, portanto, tanto os indivíduos de uma classe quanto de outra poderão ser fornecidos ao usuário como possibilidade de escolha para a realização do alinhamento local duplo.

Assim, após todas as buscas e análises realizadas sobre a ontologia, são exibidos ao usuário (em forma de uma listagem) todos os algoritmos, *Web Services* ou aplicações específicas que foram encontrados para cada uma das tarefas selecionadas para compor o *workflow*. Neste caso, as tarefas T1, T2, T3 e T5. Neste passo, o usuário poderá definir uma das possibilidades listadas (algoritmos/*Web Services* ou aplicações) para cada uma das tarefas.

– **Passo 3**: Geração de um arquivo especificado em XML, com o detalhamento das tarefas que deverão compor o *workflow* científico.

Após a definição do algoritmo, *Web Service* ou aplicação definido pelo usuário no **Passo 2** para cada uma das tarefas do *workflow*, como um último passo, a

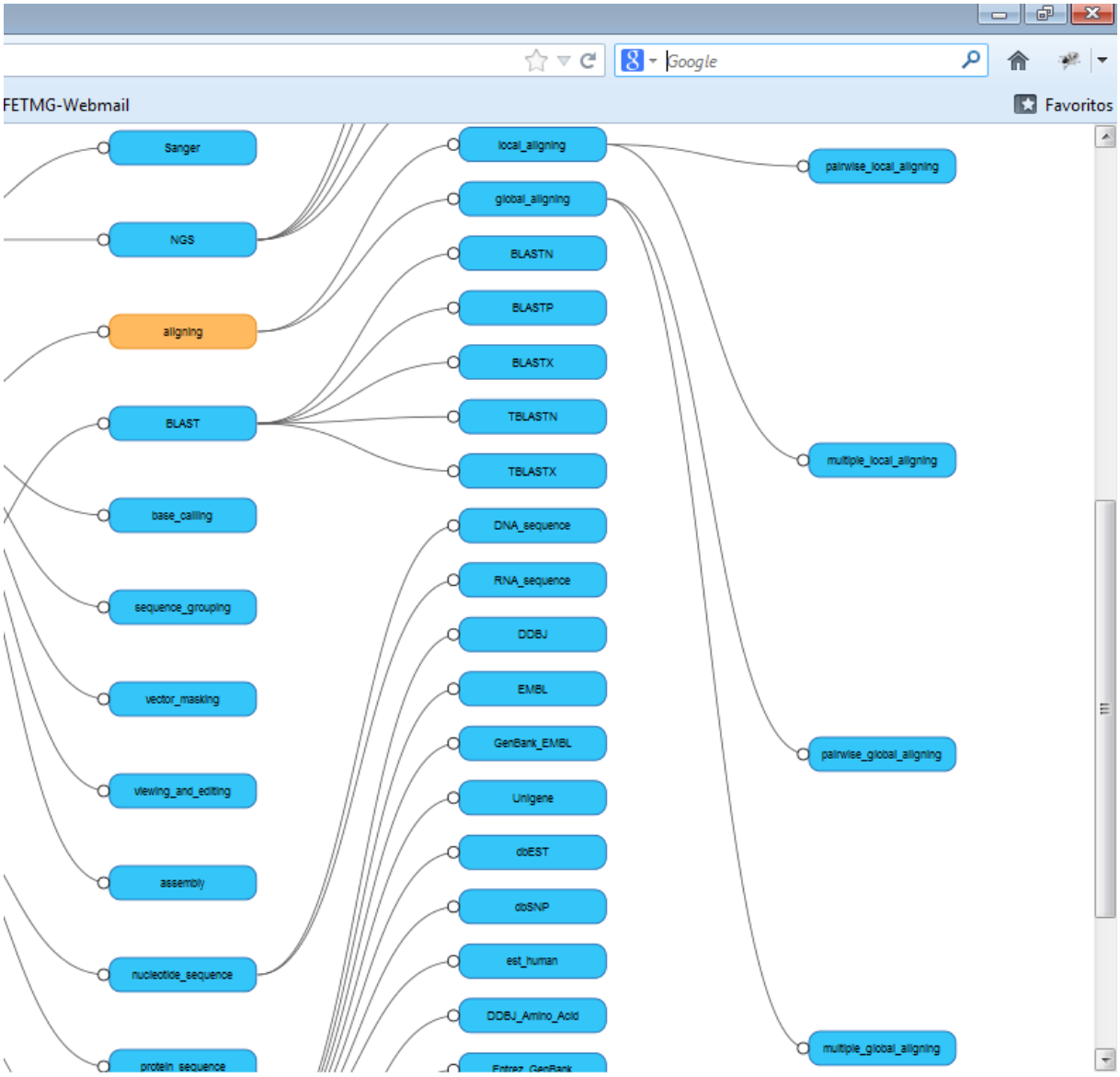

Figura 4.46: Detalhamento da tarefa de Alinhamento

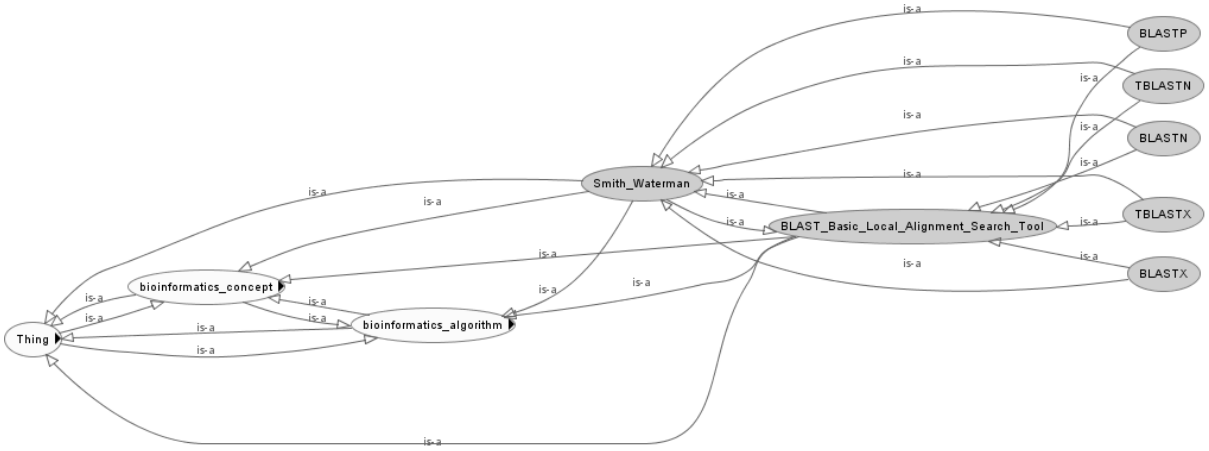

Figura 4.47: Inferência sobre a Classe *BLAST*

abordagem PL-Science gera um arquivo especificado em XML, que apresenta o encadeamento dos algoritmos, *Web Services* ou aplicações de acordo com as tarefas que deverão ser realizadas. No caso deste exemplo, a Figura 4.48 apresenta o arquivo gerado pela aplicação.

```
<Workflow>
    <Processes>
        <Process name="INB-dev:genome.imim.es:runPhred" id="Process_1">
            <Parameters>
                <Parameter name="type" value="WebService" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/2290"/>
            </Parameters>
        </Process>
        <Process name="runCrossMatchToScreenVectorService" id="Process_2">
            <Parameters>
                <Parameter name="type" value="WebService" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/1586<"/>
            </Parameters>
        </Process>
        <Process name="CAP3_Sequence_Assembly_Program" id="Process_3">
            <Parameters>
                <Parameter name="type" value="Application" />
                <Parameter name="hasLocation" value="http://pbil.univ-lyon1.fr/cap3.php"/>
            </Parameters>
        </Process>
        <Process name="NCBI-BLASTN" id="Process_4">
            <Parameters>
                <Parameter name="type" value="Application" />
                <Parameter name="hasLocation" value="http://www.biocatalogue.org/services/2268"/>
            </Parameters>
        </Process>
    </Processes>
    <Connections>
        <Connection name="Connection_1" source="Process_1" target=" Process_2"/>
        <Connection name="Connection_2" source="Process_2" target="Process_3"/>
        <Connection name="Connection_3" source="Process_3" target="Process_4"/>
    </Connections>
</Workflow>
```

Figura 4.48: Arquivo XML gerado pela Aplicação (Produto 3)

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a implementação da abordagem PL-Science em um domínio específico, neste caso, o de sequenciamento/alinhamento genético. Os modelos específicos deste domínio foram apresentados, bem como a aplicação das fases propostas na metodologia da PL-Science (Seção 3.4).

Este capítulo também ajudou a mostrar que é possível, através da abordagem PL-Science, a geração de produtos. Estes produtos podem ser classificados como uma aplicação específica ou como um *workflow* científico composto por diferentes aplicações e/ou algoritmos. A geração destes produtos pode ser considerada como uma das contribuições

da abordagem proposta.

Os produtos criados serviram também para mostrar que diferentes produtos podem ser gerados a partir de um mesmo modelo abstrato de *workflow*, sendo que cada um destes produtos possui suas particularidades. Por exemplo: o Produto 1 e o Produto 3 são bastante semelhantes com relação às tarefas que desempenham (o Produto 3 possui apenas uma tarefa além do Produto 1), porém, algumas destas tarefas são realizadas por *Web Services* diferentes e cada um dos produtos possui uma finalidade específica. Enquanto o Produto 1 tem como foco apenas o processo de sequenciamento e montagem da nova sequência obtida, o Produto 3 busca realizar comparações entre a nova sequência gerada e sequências já existentes. Já o Produto 2, que é uma aplicação isolada, também engloba a tarefa de alinhamento, como o Produto 3, porém, tem como foco o alinhamento de múltiplas sequências. Mesmo possuindo as particularidades anteriormente descritas, por outro lado, os três produtos compartilham o mesmo repositório de artefatos, e foram gerados a partir de um mesmo modelo de *features*, de uma mesma ontologia de domínio e de um mesmo modelo abstrato de *workflow*. Isto se tornou possível, pois, através do uso de diferentes modelos de domínio, conseguiu-se descrever as especificidades do domínio de sequenciamento/alinhamento. Além disso, o uso da ontologia possibilitou agregar as principais restrições relacionadas às principais tarefas do domínio científico em questão, já que estas requerem um grau maior de detalhamento, de forma a facilitar a geração dos produtos da LPSC.

A partir dos exemplos apresentados, mostrou-se também o quão necessário foi a utilização do modelo abstrato de *workflow*, proposto na Seção 4.2.3, já que este é utilizado como um 'guia' no processo de definição dos produtos da LPSC. É justamente a utilização deste modelo abstrato de *workflow* que diferencia uma Linha de Produtos de Software de uma Linha de Produtos de Software Científico, conforme proposto neste trabalho.

Deve-se ressaltar também que, através da utilização da abordagem PL-Science, conseguiu-se minimizar os problemas relacionados à questão de volatilidade de *workflows* científicos, já que, percebendo-se a necessidade de um novo *workflow* a ser gerado, este poderá ser facilmente criado usando a aplicação desenvolvida para dar suporte à LPSC. Todo este processo foi facilitado através do uso do modelo abstrato de *workflow*, que possibilita que diferentes aplicações/*workflows* possam ser criados, desde que estejam de acordo com o proposto no modelo abstrato desenvolvido.

Com relação aos pontos que precisam ser aprimorados, considerando a implementação da abordagem, pode-se citar:

- Como o mapeamento existente entre a ontologia e o modelo de *features* foi criado manualmente, com o auxílio de um especialista de domínio, tem-se como proposta futura a utilização de técnicas de alinhamento de ontologias, adaptadas para serem utilizadas com modelos de *features*, de forma a possibilitar um mapeamento semiautomático. A possibilidade de um mapeamento completamente automático foi excluída já que o auxílio de um especialista no domínio é essencial nesta etapa, de forma que não ocorra nenhum erro neste mapeamento que venha a comprometer os produtos gerados na LPSC.

- Há ainda a necessidade de acrescentar outras características relacionadas aos *Web Services*/algoritmos/aplicações que compõem o núcleo de artefatos da LPSC, de forma a aprimorar o *workflow* a ser gerado na etapa de desenvolvimento do produto, permitindo que este *workflow* possa ser executado em um Sistema de Gerenciamento de *Workflow*.

Através dos exemplos de produtos gerados a partir da LPSC desenvolvida e de acordo com a hipótese levantada neste trabalho, foi apresentado que modelos de *features* em conjunto com ontologias facilitaram o processo de criação de experimentos científicos, representados através de *workflows*, além de possibilitar o reúso de modelos e aplicações desenvolvidas para um determinado domínio.

5 CONCLUSÕES

É crescente a necessidade de busca por aplicações e/ou sistemas que auxiliem os pesquisadores em suas pesquisas/experimentos. No caso de aplicações científicas para uma determinada área, é vantajoso desenvolver uma coleção de produtos de software relacionados, utilizando uma abordagem de Linha de Produtos de Software (LPS), já que grande parte dos usuários de *frameworks* científicos normalmente começa uma nova aplicação copiando uma já existente e simplesmente ajustando-a às suas próprias necessidades. Deve-se ressaltar também as dificuldades em especificar experimentos científicos e a necessidade de um suporte semântico mais adequado para a fase de análise de domínio para a criação destes experimentos, representados através de *workflows*.

A abordagem PL-Science descrita neste trabalho utilizou modelos de *features* associados a ontologias como uma forma de facilitar a seleção e a estruturação de *workflows* científicos no contexto de uma Linha de Produtos de Software Científico (LPSC). Considerando os objetivos específicos apresentados na Seção 1.3, pode-se destacar como contribuições deste trabalho:

(i) A arquitetura, modelos de domínio e metodologia da abordagem criada para a implementação de uma LPS voltada para aplicações científicas;

(ii) Uma nova abordagem mais semântica para a especificação das variabilidades de uma LPS (usando ontologias em conjunto com modelos de *features*);

(iii) A implementação de LPSC para sequenciamento/alinhamento genético, de forma a detalhar e exemplificar a aplicação da abordagem PL-Science.

Levando em consideração a hipótese levantada neste trabalho, através da implementação da abordagem PL-Science, foram apresentados exemplos, como uma forma de prova de conceito, em que modelos de *workflows* científicos podem ser criados através da utilização de uma LPSC, através do apoio de modelos de *features* associados a ontologias, de forma a facilitar o desenvolvimento de experimentos. Portanto, através da implementação da abordagem apresentada (detalhada no Capítulo 4), pode-se inicialmente considerar que os cientistas podem seguir os modelos que especificam a LPS e tomar decisões de acordo com suas necessidades, com o objetivo de gerar um produto (uma aplicação específica ou um *workflow* científico composto por aplicações e/ou algoritmos instanciados).

No decorrer do desenvolvimento do trabalho, as principais dificuldades encontradas relacionaram-se ao conhecimento relacionado ao domínio científico escolhido. Para minimizar esta dificuldade, contou-se com o apoio de especialistas em Bioinformática, de forma a facilitar a criação dos modelos da LPSC proposta como exemplo de aplicação da abordagem PL-Science.

Como trabalhos futuros, podem ser citados diversos pontos de melhoria desta primeira versão da abordagem PL-Science, tais como:

- O desenvolvimento do Gerente Arquitetural proposto na Arquitetura da LPSC, tendo este o objetivo de possibilitar que arquiteturas diferentes possam ser facilmente utilizadas e adaptadas de acordo com as aplicações a serem desenvolvidas em um determinado domínio.

- A criação de um mapeamento semiautomático entre a ontologia e o modelo de *features*, de forma que este mapeamento não tenha a necessidade de ser realizado de forma completamente manual, como nesta versão da abordagem.

- O aprimoramento da forma de busca dos artefatos para a geração dos produtos, já que nesta versão estes artefatos são descritos como indivíduos na própria ontologia de domínio.

- O acréscimo de outras categorias de características relacionadas aos *Web Services*/algoritmos/aplicações que poderão compor o produto final, de forma a aprimorar o arquivo XML gerado ao final do processo de criação do *workflow*, tendo como objetivo final facilitar o processo de implementação do *workflow* a ser executado em um Sistema de Gerenciamento de *Workflow*.

REFERÊNCIAS

AHN, H.; KANG, S. Analysis of software product line architecture representation mechanisms. In: **Proceedings of the 2011 Ninth International Conference on Software Engineering Research, Management and Applications**, 2011. (SERA '11), p. 219–226. ISBN 978-0-7695-4490-8. Disponível em: <http://dx.doi.org/10.1109/SERA.2011.22>.

AJILA, S. A.; TIERNEY, P. J. The foom method - modeling software product line in industrial settings. In: **International Conference on Software Engineering Research and Practice**, 2002. (SERP '02).

ALMEIDA, E. **RiDE - The RiSE Process for Domain Engineering**, 2009. ISBN 3639164423, 9783639164428.

ANDERSEN, N.; CZARNECKI, K.; SHE, S.; WaSOWSKI, A. Efficient synthesis of feature models. In: **Proceedings of the 16th International Software Product Line Conference - Volume 1**, 2012. (SPLC '12), p. 106–115. ISBN 978-1-4503-1094-9. Disponível em: <http://doi.acm.org/10.1145/2362536.2362553>.

ARBEX, W. **Modelos computacionais para identificação de informação genômica associada à resistência ao carrapato bovino**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2009.

ASIKAINEN, T.; MäNNISTö, T.; SOININEN, T. Kumbang: A domain ontology for modelling variability in software product families. **Adv. Eng. Inform.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, n. 1, p. 23–40, jan. 2007. ISSN 1474-0346. Disponível em: <http://dx.doi.org/10.1016/j.aei.2006.11.007>.

ATKINSON, C.; BAYER, J.; MUTHIG, D. Component-based product line development: the kobra approach. In: **Proceedings of the first conference on Software product lines : experience and research directions: experience and research directions**, 2000. p. 289–309. ISBN 0-79237-940-3. Disponível em: <http://dl.acm.org/citation.cfm?id=355461.357556>.

BAYER, J.; FLEGE, O.; KNAUBER, P.; LAQUA, R.; MUTHIG, D.; SCHMID, K.; WIDEN, T.; DEBAUD, J.-M. Pulse: a methodology to develop software product lines. In: **Proceedings of the 1999 symposium on Software reusability**, 1999. (SSR '99), p. 122–131. ISBN 1-58113-101-1. Disponível em: <http://doi.acm.org/10.1145/303008.303063>.

BECHHOFER, S.; HARMELEN, F. van; HENDLER, J.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A. **OWL Web Ontology Language Reference**, February 2004.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. **Scientific American**, v. 284, n. 5, p. 34–43, maio 2001. Disponível em: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>.

BEUCHE, D. Modeling and building software product lines with pure: : variants. In: SCHAEFER, I.; JOHN, I.; SCHMID, K. (Ed.). **Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22-26, 2011. Proceedings**, 2011. p. 46. ISBN 978-1-4503-0789-5.

BHAGAT, J.; TANOH, F.; NZUOBONTANE, E.; LAURENT, T.; ORLOWSKI, J.; ROOS, M.; WOLSTENCROFT, K.; ALEKSEJEVS, S.; STEVENS, R.; PETTIFER, S.; LOPEZ, R.; GOBLE, C. A. BioCatalogue: a universal catalogue of web services for the life sciences. **Nucleic acids research**, Article in press, maio 2010. Disponível em: <http://dx.doi.org/10.1093/nar/gkq394>.

CAP3. **CAP3 Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://pbil.univ-lyon1.fr/cap3.php>.

CARVALHO, A.; BRAYNER, A.; LOUREIRO, A.; FURTADO ANTONIO L., e. a. **Grandes Desafios da Pesquisa em Computação no Brasil (2006-2016)**, 2006.

CARVALHO, M. C. C. G.; SILVA, D. C. G. Sequenciamento de dna de nova geração e suas aplicações na genômica de plantas. 2010.

CIRILO, E.; KULESZA, U.; LUCENA, C. Genarch a model-based product derivation tool. In: **Proceedings of the First Brazilian Symposium on Components, Architecture and Reuse (SBCARS 2007)**, 2007.

CLEMENTS, P.; NORTHROP, L. **Software product lines: practices and patterns**, 2001. ISBN 0-201-70332-7.

COSTA, G. C. B.; BRAGA, R. M. M.; DAVID, J. M. N. Apoiando uma linha de produtos de workflow científico através da associação de ontologias e modelos de features. In: **II Workshop de Teses e Dissertações do CBSoft**, 2012. (CBSoft '12), p. 1–5.

COSTA, G. C. B.; BRAGA, R. M. M.; DAVID, J. M. N.; CAMPOS, F. Connecting feature models and ontologies in software product lines. In: MACIASZEK, L. A.; CUZZOCREA, A.; CORDEIRO, J. (Ed.). **ICEIS (2)**, 2012. p. 160–163. ISBN 978-989-8565-11-2. Disponível em: <http://dblp.uni-trier.de/db/conf/iceis/iceis2012-2.htmlCostaBDC12>.

CROSS_MATCH. **Cross_match Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://www.incogen.com/public_documents/vibe/details/crossmatch.html>.

CZARNECKI, K.; KIM, C. H. **Cardinality-Based Feature Modeling and Constraints: A Progress Report**, out. 2005.

CZARNECKI, K.; KIM, C. H. P.; KALLEBERG, K. T. Feature models are views on ontologies. In: **Proceedings of the 10th International on Software Product Line Conference**, 2006. (SPLC '06), p. 41–51. ISBN 0-7695-2599-7. Disponível em: <http://dl.acm.org/citation.cfm?id=1158337.1158678>.

DARNELL J. E., L. H. F. B. D. **Molecular Cell Biology**, 1986.

DEELSTRA, S.; SINNEMA, M.; BOSCH, J. Product derivation in software product families: a case study. **J. Syst. Softw.**, Elsevier Science Inc., New York, NY, USA, v. 74, n. 2, p. 173–194, jan. 2005. ISSN 0164-1212. Disponível em: <http://dx.doi.org/10.1016/j.jss.2003.11.012>.

EMBL-EBI. **The European Bioinformatics Institute**. 2012. Acesso em 20 dez. 2012. Disponível em: <http://www.ebi.ac.uk>.

EMBOSS. **Abiview**. 2013. Acesso em 15 jan. 2013. Disponível em: <http://emboss.bioinformatics.nl/cgi-bin/emboss/abiview>.

EWING, B.; HILLIER, L.; WENDL, M.; GREEN, P. Base-calling of automated sequencer traces using Phred. I. Accuracy assessment. **Genome Research**, COLD SPRING HARBOR LAB PRESS, 1 BUNGTOWN RD, PLAINVIEW, NY 11724 USA, 8, n. 3, p. 175–185, mar. 1998. ISSN 1054-9803.

FACT++. **FaCT++ Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://owl.man.ac.uk/factplusplus/>.

FERNANDES, P.; WERNER, C.; MURTA, L. G. P. Feature modeling for context-aware software product lines. In: **SEKE**, 2008. p. 758–763.

FILHO, J. a. B. F.; BARAIS, O.; BAUDRY, B.; VIANA, W.; ANDRADE, R. M. C. An approach for semantic enrichment of software product lines. In: **Proceedings of the 16th International Software Product Line Conference - Volume 2**, 2012. (SPLC '12), p. 188–195. ISBN 978-1-4503-1095-6. Disponível em: <http://doi.acm.org/10.1145/2364412.2364444>.

GOBLE, C. A.; ROURE, D. D. The impact of workflow tools on data-centric research. In: HEY, T.; TANSLEY, S.; TOLLE, K. M. (Ed.). **The Fourth Paradigm**, 2009. p. 137–145. ISBN 978-0982544204. Disponível em: <http://dblp.uni-trier.de/db/books/collections/4paradigm2009.htmlGobleR09>.

GOMAA, H. **Designing software product lines with UML - from use cases to pattern-based software architectures**, 2005. I-XXVII, 1-701 p. ISBN 978-0-201-77595-2.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. **Int. J. Hum.-Comput. Stud.**, Academic Press, Inc., Duluth, MN, USA, v. 43, n. 5-6, p. 907–928, dez. 1995. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1006/ijhc.1995.1081>.

GUARINO, N. **Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy**. 1st. ed., 1998. ISBN 9051993994.

HOFMEISTER, C.; NORD, R.; SONI, D. **Applied software architecture**, 2000. ISBN 0-201-32571-3.

HORRIDGE, M.; BRANDT, S. **A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.3**, 2011. 108 p.

ITANA; TRAVASSOS, G. H. O Enfoque de Linha de Produto para Desenvolvimento de Software. In: aO, S. B. de C. (Ed.). **XXI JAI - Jornadas de Atualização em Informática - Livro Texto**, 2002. p. 1–32.

JOHANSEN, M. F.; FLEUREY, F.; ACHER, M.; COLLET, P.; LAHIRE, P. Exploring the synergies between feature models and ontologies. In: **SPLC Workshops**, 2010. p. 163–170.

JUNIOR, E. A. O.; MALDONADO, J. C.; GIMENES, I. M. S. Empirical validation of complexity and extensibility metrics for software product line architectures. In: **Proceedings of the 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse**, 2010. (SBCARS '10), p. 31–40. ISBN 978-0-7695-4259-1. Disponível em: <http://dx.doi.org/10.1109/SBCARS.2010.13>.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**, November 1990.

KANG, K. C.; KIM, S.; LEE, J.; KIM, K.; SHIN, E.; HUH, M. Form: A feature-oriented reuse method with domain-specific reference architectures. **Ann. Softw. Eng.**, J. C. Baltzer AG, Science Publishers, Red Bank, NJ, USA, v. 5, p. 143–168, jan. 1998. ISSN 1022-7091. Disponível em: <http://dl.acm.org/citation.cfm?id=590631.590645>.

KRUEGER, C. W. The biglever software gears systems and software product line lifecycle framework. In: **SPLC Workshops**, 2010. p. 297.

LEE, J.; MUTHIG, D. Feature-oriented variability management in product line engineering. **Commun. ACM**, ACM, New York, NY, USA, v. 49, n. 12, p. 55–59, dez. 2006. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1183236.1183266>.

LEMOS, M. **Workflow para Bioinformática**. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, 2004.

LINDEN, F. J. v. d.; SCHMID, K.; ROMMES, E. **Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering**, 2007. ISBN 3540714367.

LIVENGOOD, S. Issues in software product line evolution: complex changes in variability models. In: **Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering**, 2011. (PLEASE '11), p. 6–9. ISBN 978-1-4503-0584-6. Disponível em: <http://doi.acm.org/10.1145/1985484.1985487>.

MATOS, E. E. **Um Framework Baseado em Ontologias com Serviços Web para Modelagem Conceitual em Biologia Sistêmica**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2008.

MATTOS, A.; SILVA, F. C.; RUBERG, N.; CRUZ, S. M. S.; MATTOSO, M. L. Q. **Gerência de Workflows Científicos: Uma Análise Crítica no Contexto da Bioninformática**, 2008.

MATTOSO, M.; WERNER, C.; TRAVASSOS, G. H.; BRAGANHOLO, V.; OGASAWARA, E.; OLIVEIRA, D. D.; CRUZ, S. M.; MARTINHO, W.; MURTA, L. Towards supporting the life cycle of large scale scientific experiments. **International Journal of Business Process Integration and Management**, v. 5, n. 1, p. 79+, 2010. ISSN 1741-8763. Disponível em: <http://dx.doi.org/10.1504/ijbpim.2010.033176>.

MIRA. **MIRA Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://www.molecularevolution.org/software/genomics/mira>.

NARDI, A. R. **Uma arquitetura de baixo acoplamento para execução de padrões de controle de fluxo em grades**. Tese (Doutorado) — Universidade de São Paulo, 2009.

ODYSSEY. **Odyssey SDE Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://reuse.cos.ufrj.br/odyssey>.

OGASAWARA, E.; PAULINO, C.; MURTA, L.; WERNER, C.; MATTOSO, M. Experiment line: Software reuse in scientific workflows. In: WINSLETT, M. (Ed.). **Scientific and Statistical Database Management**, 2009, (Lecture Notes in

Computer Science, v. 5566). p. 264–272. ISBN 978-3-642-02278-4. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02279-1$_2$0>.

OLIVEIRA, D. de; OGASAWARA, E. S.; CHIRIGATI, F. S.; SILVA, V.; MURTA, L. G. P.; MATTOSO, M. Gexpline: A tool for supporting experiment composition. In: MCGUINNESS, D. L.; MICHAELIS, J.; MOREAU, L. (Ed.). **IPAW**, 2010. (Lecture Notes in Computer Science, v. 6378), p. 251–259. ISBN 978-3-642-17818-4. Disponível em: <http://dblp.uni-trier.de/db/conf/ipaw/ipaw2010.htmlOliveiraOCSMM10>.

PHRAP. **Phrap Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://www.phrap.org/>.

POHL, K.; BöCKLE, G.; LINDEN, F. J. v. d. **Software Product Line Engineering: Foundations, Principles and Techniques**, 2005. ISBN 3540243720.

PROSDOCIMI, F.; CERQUEIRA, G.; INNECK, E.; SILVA, A.; REIS, A.; JUNQUEIRA, A.; SANTOS, A.; NHANI-JÚNIOR, A.; WUST, C.; CAMARGO-FILHO, F.; KESSED-JIAN, J.; PETRETSKI, J.; CAMARGO, L.; FERREIRA, R.; LIMA, R.; PEREIRA, R.; JARDIM, S.; SAMPAIO, V.; FOLGUERAS-FLATSCHART, A. V. Bioinformática: Manual do usuário. **Biotecnologia Ciência e Desenvolvimento**, v. 29 - novembro/-decembro, p. 12–25, 2002.

REINHARTZ-BERGER, I.; STURM, A. Utilizing domain models for application design and validation. **Inf. Softw. Technol.**, Butterworth-Heinemann, Newton, MA, USA, v. 51, n. 8, p. 1275–1289, ago. 2009. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2009.03.005>.

REINHARTZ-BERGER, I.; STURM, A.; TSOURY, A. Comprehension and utilization of core assets models in software product line engineering. In: **CAiSE Forum**, 2011. p. 171–178.

REMMEL, H.; PAECH, B.; ENGWER, C.; BASTIAN, P. Supporting the testing of scientific frameworks with software product line engineering: a proposed approach. In: **Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering**, 2011. (SECSE '11), p. 10–18. ISBN 978-1-4503-0598-3. Disponível em: <http://doi.acm.org/10.1145/1985782.1985785>.

RIEDEL, M.; STREIT, A.; WOLF, F.; LIPPERT, T.; KRANZLMULLER, D. Classification of different approaches for e-science applications in next generation computing infrastructures. In: **eScience, 2008. eScience '08. IEEE Fourth International Conference on**, 2008. p. 198–205.

SANGER, F.; NICKLEN, S.; COULSON, A. Dna sequencing with chain-terminating inhibitors. **Proceedings of the National Academy of Sciences of the United States of America (PNAS)**, v. 74, p. 5463–5467, 1977.

SILVA, A. P. **Uma Linha de Produto de Software baseada na Web Semântica para Sistemas Tutores Inteligentes**. Tese (Doutorado) — Universidade Federal de Campina Grande, 2011.

SILVA, C. E. P. **Captura de Dados de Proveniência de Workflows Científicos em Nuvens Computacionais**. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2011.

SILVA, L. A. M. **Composer-Science: Um Framework para a Composição de Workflows Científicos**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2010.

SILVA, L. M. da; BRAGA, R. M. M.; CAMPOS, F. Composição de workflows científicos em projetos de e-science. In: **SBES**, 2011. p. 273–282.

SILVA, L. M. da; BRAGA, R. M. M.; CAMPOS, F. Composer-science: A semantic service based framework for workflow composition in e-science projects. **Inf. Sci.**, v. 186, n. 1, p. 186–208, 2012.

SLETHOLT, M. T.; HANNAY, J.; PFAHL, D.; BENESTAD, H. C.; LANGTANGEN, H. P. A literature review of agile practices and their effects in scientific software development. In: **Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering**, 2011. (SECSE '11), p. 1–9. ISBN 978-1-4503-0598-3. Disponível em: <http://doi.acm.org/10.1145/1985782.1985784>.

SOUZA, N. B. O. **Caracterização de Software Científico: Um Estudo de Caso em Modelagem Computacional**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2011.

STEIN, L. D. An Introduction to the Informatics of Next-Generation Sequencing. **Current Protocols in Bioinformatics**, John Wiley & Sons, Inc., Chapter 11, dez. 2002. ISSN 1934-340X. Disponível em: <http://dx.doi.org/10.1002/0471250953.bi1101s36>.

SVAHNBERG, M.; GURP, J. van; BOSCH, J. A taxonomy of variability realization techniques: Research articles. **Softw. Pract. Exper.**, John Wiley & Sons, Inc., New York, NY, USA, v. 35, n. 8, p. 705–754, jul. 2005. ISSN 0038-0644. Disponível em: <http://dx.doi.org/10.1002/spe.v35:8>.

TECHNELYSIUM. **Chromas**. 2013. Acesso em 15 jan. 2013. Disponível em: <http://www.technelysium.com.au/chromas.html>.

VELVET. **VELVET Homepage**. 2013. Acesso em 28 fev. 2013. Disponível em: <http://www.molecularevolution.org/software/genomics/velvet>.

WOLSTENCROFT, K.; ALPER, P.; HULL, D.; WROE, C.; LORD, P. W.; STEVENS, R. D.; GOBLE, C. A. The mygrid ontology: bioinformatics service discovery. **Int. J. Bioinformatics Res. Appl.**, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 3, n. 3, p. 303–325, set. 2007. ISSN 1744-5485. Disponível em: <http://dx.doi.org/10.1504/IJBRA.2007.015005>.

YU, W.; SMITH, S. Reusability of fea software: A program family approach. In: **Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering**, 2009. (SECSE '09), p. 43–50. ISBN 978-1-4244-3737-5. Disponível em: <http://dx.doi.org/10.1109/SECSE.2009.5069161>.