UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA O PROJETO DE APLICAÇÕES WEB DE INTEGRAÇÃO DE DADOS

Dissertação de Mestrado

TICIANNE DE GOIS RIBEIRO

Fortaleza, janeiro de 2011.

UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA O PROJETO DE APLICAÇÕES WEB DE INTEGRAÇÃO DE DADOS

Dissertação apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação.

Orientadora:

Prof.ª. Dra. Vânia Maria Ponte Vidal

Ticianne de Gois Ribeiro

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA O PROJETO DE APLICAÇÕES WEB DE INTEGRAÇÃO DE DADOS

Ticianne de Gois Ribeiro

Dissertação apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação.

Composição da Banca Examinadora:

_____

Prof.ª. Dra. Vânia Maria Ponte Vidal (DC/UFC)

Orientador

_____

Prof. Dr. Fábio André Machado Porto (LNCC/RJ)

_____

Prof. Dr. José Maria Monteiro (DC/UFC)

_____

Prof. Dr. José Antônio Fernandes de Macêdo (DC/UFC)

Aprovada em 26 de janeiro de 2011.

À minha família.

AGRADECIMENTOS

Ao Autor e Consumador da minha fé, sem o qual eu jamais teria tido um dia sequer de vida. Nenhuma conquista obtida veio de minha própria força ou sabedoria, mas da graça e misericórdia com a qual Ele tem me sustentado todos os dias de minha vida.

Aos meus pais Ivaldo e Jaqueline, pelo amor constante e pelo incentivo a sempre ir mais longe; pelo suporte que me deram sempre que precisei, não só para concluir este trabalho, mas para todos os momentos. Ao meu irmão Thiago, companheiro à sua forma, muito apreciada por mim.

Ao meu noivo Daniel, pelo apoio, amor, carinho e suporte. Pela cobrança e pelas palavras amigas sempre chegadas em boa hora. Pelo constante incentivo para seguir em frente.

À minha muito querida orientadora Vânia, pela dedicação, paciência, ensinamentos. Por muitas lições aprendidas para sempre, por acreditar em mim e me cobrar sempre o melhor, fazendo com que eu mesma fosse também melhor. Pela força e pela confiança que jamais serão esquecidas.

Aos professores José Macêdo, José Maria e Fábio Porto, por terem aceitado e convite e por contribuírem, em diversos momentos e de diversas formas com a melhoria deste trabalho.

Aos meus queridos companheiros de mestrado, por todos os momentos de estudo, ajuda e risadas. Este trabalho de certa forma é conquista de todos vocês também: Ticiana, Régis, Macedo, Lívia, Bruno, Fernando Wagner, Clayton, Thiago, Hélio, Débora, Cadu e Mônica.

Às minhas amadas amigas Raquel, Karine e Camila, que sempre estiveram dispostas a me escutar e tentar ajudar.

Aos funcionários do Departamento de Computação, especialmente ao Orley e a Rosana, que sempre estiveram prontos para nos ajudar.

A todos os meus familiares e amigos que se alegraram comigo pela conquista deste objetivo.

A Universidade Federal do Ceará (UFC) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), por tornarem possível a realização deste mestrado.

SUMÁRIO

Lista de Figuras

Resumo

Abstract

1. INTRODUÇÃO 9

1.1 DESCRIÇÃO DO PROBLEMA E MOTIVAÇÃO 9
1.2 DATA-INTENSIVE WEB APLICATIONS 13
1.3 APLICAÇÕES MASHUP 14
1.4 DESCRIÇÃO DA PROPOSTA E CONTRIBUIÇÕES 17
1.5 TRABALHOS RELACIONADOS 20
1.6 ORGANIZAÇÃO DA DISSERTAÇÃO 22

2. INTEGRAÇÃO SEMÂNTICA 23

2.1 INTEGRAÇÃO DE DADOS 23
2.2 ARQUITETURAS DE INTEGRAÇÃO SEMÂNTICA 25
2.2.1 Arquitetura de Dois Níveis de Ontologias 25
2.2.2 Linked Data 26
2.3 REPRESENTAÇÃO DO ESQUEMA CONCEITUAL DE APLICAÇÕES WEB ATRAVÉS DE ONTOLOGIAS 29
2.4 REPRESENTAÇÃO DE FONTES DE DADOS ATRAVÉS DE ONTOLOGIAS 30
2.5 INFRAESTRUTURA BASEADA EM ONTOLOGIAS PARA ESPECIFICAÇÃO DE APLICAÇÕES WEB DE INTEGRAÇÃO DE DADOS 33
2.6 CONSIDERAÇÕES FINAIS 35

3. MÉTODO PARA PROJETO DE APLICAÇÕES WEB 36

3.1 VISÃO GERAL 36
3.2 ETAPAS DO MÉTODO 38
3.3 PRÉ-REQUISITO: INTEGRAÇÃO SEMÂNTICA 40
3.3.1 Passo 1: Definição da Ontologia de Aplicação (OA) 41
3.3.2 Passo 2: Geração das Ontologias Exportadas 42
3.3.3 Passo 3: Definição de Mapeamentos OA-OE 43
3.4 CONSIDERAÇÕES FINAIS 44

4. ESPECIFICAÇÃO CONCEITUAL E DEFINIÇÃO DA APLICAÇÃO 45

4.1 APLICAÇÃO EXEMPLO 45
4.2 ETAPA 1: ESPECIFICAÇÃO CONCEITUAL DA APLICAÇÃO 48
4.2.1 Passo 1: Definição de Interações do Usuário 51

4.2.2 Passo 2: Definição de Esquema ($\mathcal{S}$), Parâmetros ($\mathcal{P}$) e Elos ($\mathcal{E}$) 53
4.2.3 Passo 3: Geração Automática da Consulta Q 55
4.3 ETAPA 2: DEFINIÇÃO DOS PLANOS DE EXECUÇÃO 59
4.4 CONSIDERAÇÕES FINAIS 62

5. PUBLICAÇÃO DOS DADOS 63

5.1 ANOTAÇÃO E PUBLICAÇÃO DE VCDS 63
5.2 SERVIÇOS WEB DE ACESSO A DADOS 64
5.3 ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB 66
5.4 ANOTAÇÃO E PUBLICAÇÃO DE VCDS COMO SERVIÇOS WEB DE ACESSO A DADOS 68
5.5 CENÁRIOS DE ACESSO À APLICAÇÃO WEB 69
5.6 CONSIDERAÇÕES FINAIS 72

6. FRAMEWORK CONCEITUAL E PROTÓTIPO DE FERRAMENTA 73

6.1 VISÃO GERAL DO FRAMEWORK CONCEITUAL 73
6.1.1 Módulos de Integração Semântica 75
6.1.2 Módulos de Especificação Conceitual da Aplicação 76
6.1.3 Módulos de Definição dos Planos de Execução 76
6.1.4 Módulos de Publicação de Dados 77
6.1.5 Repositório de Artefatos 78
6.2 PROTÓTIPO DE FERRAMENTA PARA SUPORTE AO MÉTODO 79
6.2.1 Projeto da Interface 79
6.3 IMPLEMENTAÇÃO DO PROTÓTIPO 82
6.4 CONSIDERAÇÕES FINAIS 85

7. DISCUSSÃO FINAL 86

7.1 CONCLUSÕES 86
7.2 SUGESTÕES PARA TRABALHOS FUTUROS 88

8. REFERÊNCIAS BIBLIOGRÁFICAS 89

9. APÊNDICE A – ESTUDO DE CASO 90

A.1 INTEGRAÇÃO SEMÂNTICA 91
A2. ESPECIFICAÇÃO CONCEITUAL DA APLICAÇÃO 95
A3. DEFINIÇÃO DOS PLANOS DE EXECUÇÃO DAS VCDS 97
A4. PUBLICAÇÃO DOS DADOS 100

10. APÊNDICE B – ALGORITMO PARA GERAÇÃO DE CONSULTAS 101

11. APÊNDICE C – PROJETO DA FERRAMENTA 104

C1. FUNCIONALIDADES DA CAMADA DE INTEGRAÇÃO SEMÂNTICA 104
C.2 FUNCIONALIDADES DA CAMADA DE ESPECIFICAÇÃO CONCEITUAL DA APLICAÇÃO 108
C.3 FUNCIONALIDADES DA CAMADA DE IMPLEMENTAÇÃO DA VCD 114

12. APÊNDICE D - LEVANTAMENTO DE TECNOLOGIAS 117

LISTA DE FIGURAS

1.1 Arquitetura de 3 camadas de aplicações DIWA.....14
1.2 Arquitetura de mashup.....16
1.3 Etapas do Método para projeto de aplicações Web de integração de dados 18
2.1 Arquitetura de dois níveis.....25
2.2 Datasets no Linked Data (linkeddata.org, março de 2009).....27
2.3 Infraestrutura Baseada em Ontologias para Especificação da Aplicação Web.....34
3.1 Diagrama de Atividades da etapa de Integração Semântica.....40
4.1 Exemplo de aplicação Web de integração de dados: Mashup Doenças e Drogas.....46
4.2 Ontologia da Aplicação Mashup Doenças e Drogas.....47
4.3 Atividades da etapa de Especificação Conceitual.....50
4.4 Exemplo de definição do XML Schema de uma VCD.....50
4.5 UID “drogasPorDoença”, pertinente à aplicação D&D.....52
4.6 Schema XML padrão (S) de uma VCD.....54
4.7 Definição de S para a interação V2.....54
4.8 Schema XML padrão do arquivo de assertivas de correspondência.....56
Consulta Q : define V2 sobre a OA da aplicação Doenças&Drogas.....58
4.9 Representação da VCD drugsIndex.....58
4.10 Atividades da etapa de Definição dos Planos de Execução.....60
4.11 Plano de Consulta Federado Q3’’.....61
5.1 Passos da etapa de Publicação de Dados.....64
5.2 Esquema de busca semântica por VCDs.....69
6.1 Módulos e artefatos do Framework Conceitual.....74
6.2 Tela Inicial da ferramenta.....80
6.3 Tela da Etapa de Integração Semântica.....81
6.4 Tela da Etapa de Especificação Conceitual da VCD.....82
6.5 Diagrama de Classes da implementação da atividade de Criação da VCD.....83

RESUMO

Aplicações Web de Integração de Dados são atualmente as mais difundidas dentre as aplicações da Web. Destacam-se dois tipos: mashups, que são aplicações web de curta duração que visam combinar dados e serviços provenientes de diversas fontes; e as denominadas Data-Intensive Web Applications (DIWA), cujo principal objetivo é publicar e integrar dados dinamicamente extraídos de uma ou mais fontes de dados. Ambos os tipos de aplicações lidam com unidades de conteúdo dinâmico, isto é, conteúdo recuperado à medida que o usuário o solicita durante suas interações com a aplicação. Tais dados em geral são dinamicamente recuperados através de um conjunto de especificações de dados sobre fontes distribuídas e das definições de relacionamento entre estas fontes. Especificar, construir e manter estas especificações não são tarefas fáceis, devido à necessidade de conhecer as URIs das fontes e o vocabulário utilizado por cada fonte, além da dificuldade para encontrar relações entre dados destas fontes. O objetivo deste trabalho é definir uma metodologia que permite a especificação em alto nível deste conteúdo dinâmico de forma que, a partir desta especificação permita a geração dos planos de execução para as consultas da aplicação, seguido da publicação destes dados na Web Semântica, com base no entendimento das relações entre fontes e na integração semântica entre estas. Neste documento, é definido um método que orienta os usuários na criação de Aplicações Web de Integração de Dados, permitindo uma especificação de alto nível do conteúdo dinâmico da aplicação, sobre uma visão unificada do conjunto de dados a ser consultado.

1. Introdução

1.1 Descrição do Problema e Motivação

Aplicações Web são um dos meios mais utilizados atualmente para a disseminação e o compartilhamento de conteúdo multimídia, para a oferta de serviços e para a criação de oportunidades de negócios por organizações públicas e privadas. Este tipo de aplicação evoluiu a partir dos *websites* utilizados por estas organizações, inicialmente, apenas para a disponibilização de informações.

Um *website* trata-se de um sistema hipermídia distribuído, no qual os recursos estão interligados através de *links,* resultando em um conjunto de páginas Web relacionadas, pertencentes a um determinado domínio. Uma aplicação Web funciona como uma extensão de um website, enriquecendo-o com serviços e funcionalidades, i.e., permitindo que o usuário execute uma lógica de negócios, através de um navegador Web. Isto significa dizer que, em aplicações Web, as entradas fornecidas pelo usuário – através de navegação ou de campos de entrada de dados – afetam o estado do negócio [Conallen 2003].

Existem dois cenários possíveis para o desenvolvimento de aplicações Web, com relação à obtenção dos dados a serem manipulados pela aplicação: (i) *os dados são extraídos de um banco de dados definido exclusivamente para a aplicação*, o qual atende a todos os requisitos de dados e regras de negócio da aplicação para a qual foi desenvolvido. Este cenário caracteriza aplicações Web autocontidas com relação aos dados, i.e., todas as suas dependências de dados podem ser satisfeitas em si própria e não há necessidade de comunicação com outro sistema para obtenção de dados, senão com o banco de dados definido para a própria aplicação; (ii) *os dados são extraídos de uma ou mais fontes de dados já existentes*, por meio de tradutores que possibilitam o acesso aos dados e de um processo de integração de dados (quando necessário). Estes dados são então processados de forma a atender aos requisitos de dados e regras de negócio da aplicação que os solicita.

O segundo cenário de desenvolvimento de aplicações Web constitui o alvo deste trabalho. Tal cenário caracteriza aplicações mais complexas, cujos dados encontram-se em uma fonte desenvolvida não exclusivamente para a aplicação; ou mesmo distribuídos em fontes heterogêneas. Como exemplo de aplicações pertencentes a este cenário, é possível citar aplicações Web de reservas de hotel, as quais acessam dados dos diversos hotéis cadastrados na aplicação; e aplicações que fornecem informações de eventos associados a localizações geográficas. Neste contexto, dois tipos de aplicações Web têm atualmente se destacado por seu crescente uso: Data-Intensive Web Application (DIWA) e Mashups.

Aplicações do tipo DIWA são caracterizadas por requererem acesso intenso e dinâmico a grandes quantidades de dados, extraídos de uma ou mais fontes, cujas estruturas de armazenamento são geralmente complexas [Ceri 2003]. Por outro lado, aplicações Web do tipo Mashup combinam recursos de aplicações Web distintas para criar novas aplicações ou serviços. Ambos os tipos de aplicações Web são utilizadas para a integração de dados, provenientes de fontes heterogêneas e distribuídas. Em geral, estes dados são obtidos dinamicamente, i.e., à medida que o usuário os solicita por meio de interações com a aplicação. Entretanto, há um conjunto de características que diferencia aplicações DIWA e aplicações baseadas em Mashups, tais quais: o tipo de integração realizada [Yu et al. 2008], o volume e o tipo de dados manipulado e o modelo de composição utilizado [Lorenzo et al. 2009].

O desenvolvimento e a manutenção, tanto de aplicações DIWA quanto de Mashups, está associado a dificuldades relativas ao acesso e a integração de múltiplas fontes heterogêneas, devido às diversidades estruturais e semânticas entre os esquemas a serem integrados. Tais diversidades tornam complexa a especificação do conteúdo dinâmico a ser apresentado nas páginas destas aplicações Web.

Em aplicações DIWA, não existe uma linguagem de alto nível utilizada para a especificação dos requisitos de conteúdo das páginas da aplicação i.e., independentemente da linguagem utilizada na implementação. A definição requisitos de conteúdo sobre as fontes e a integração entre esquemas são realizados pelo desenvolvedor manualmente, através do uso de módulos compilados e scripts interpretados, e.g., JavaScript, PHP ou Ruby. Esta prática implica em um alto custo de desenvolvimento, devido a grande quantidade e complexidade de código específico da aplicação e da plataforma. Além disso, não há uma metodologia definida para a especificação dos requisitos de conteúdo das páginas de aplicações DIWA.

Este fato traz dificuldades na manutenção deste tipo de aplicação, devido à necessidade de alteração manual dos scripts que obtêm o conteúdo dinâmico das páginas. Essa situação é ainda agravada quando o conteúdo dinâmico é extraído de múltiplas fontes de dados, surgindo os problemas de interoperabilidade estrutural, uma vez que os dados são armazenados em estruturas distintas; e semântica, onde o significado das informações, presentes nas diversas bases, não possui a mesma interpretação para todas as bases.

Quanto às aplicações Mashup, tipicamente, um grande esforço de programação é necessário para a sua criação: são necessários conhecimentos, não somente de como escrever o código, mas também sobre as APIs necessárias para a invocação dos dados pertencentes a aplicações de terceiros. Há a necessidade de conhecimento sobre a implementação de técnicas de extração de conteúdo para o caso de aplicações que não disponibilizam APIs, além do conhecimento da estrutura de dados da entrada e saída de cada elemento de conteúdo acessado [Lorenzo et al. 2009].

Uma prática comum, no desenvolvimento de Mashups, é o uso de técnicas de extração de dados para a obtenção de dados diretamente das páginas das aplicações. O uso destas técnicas traz grandes dificuldades ao desenvolvedor e prejuízos à recuperação semântica dos dados, uma vez que os conteúdos criados para consumo humano não são conteúdos próprios para o consumo automático por máquinas. Portanto, a criação e a manutenção destas aplicações são processos difíceis de realizar, devido à complexidade e quantidade de código *ad-hoc* específico da aplicação e da API fornecida pelo provedor de conteúdo.

Os problemas encontrados nos cenários de desenvolvimento e utilização de Aplicações Web de Integração de Dados (tanto DIWA quanto mashups), os quais são abordados neste trabalho, podem ser enumerados da seguinte forma:

1. *Falta de um ciclo de desenvolvimento bem definido*

Atualmente, não existe uma metodologia bem definida para o desenvolvimento de aplicações Web de integração de dados. Entretanto, o uso de uma metodologia no desenvolvimento destas aplicações traria diversos benefícios, dentre os quais: aumento na qualidade da aplicação, devido à padronização; independência dos indivíduos, pois, conhecendo as etapas seguidas durante o desenvolvimento da aplicação, um desenvolvedor é capaz de realizar mais facilmente a manutenção em um sistema desenvolvido por terceiros; aumento da produtividade, uma vez que aplicações bem construídas são mais facilmente reutilizáveis; e gerenciamento eficiente do desenvolvimento, devido ao conhecimento prévio das etapas a serem seguidas e dos respectivos produtos obtidos.

2. *Dificuldade na especificação do conteúdo dinâmico*

   O projeto de aplicações de integração de dados enfrenta alguns problemas recorrentes, devido à forma como o conteúdo dinâmico, i.e, conteúdo solicitado pelo usuário, sob demanda, de acordo com um conjunto de parâmetros, é especificado: uma vez que o conteúdo deve ser especificado como consultas sobre fontes distribuídas, é necessário conhecimento específico do usuário sobre uma linguagem de consulta e/ou programação, bem como das URIs das fontes de dados e do vocabulário dessas fontes. Essa dificuldade é reforçada pela falta de uma linguagem de alto nível para consultar fontes heterogêneas. Além disso, o usuário deve lidar com diferentes APIs Web para acessar diferentes fontes e serviços [Lorenzo et al. 2009]. Finalmente, há também um problema relacionado à semântica: devido aos diversos vocabulários das fontes em questão, é difícil encontrar relações entre dados residentes nestas fontes. Dessa forma, uma questão chave para o desenvolvimento deste tipo de aplicação é como criar planos de consulta integrados, para uma aplicação de domínio específico.

3. *Dificuldade na reutilização dos dados obtidos*

   Permitir que tais aplicações disponibilizem seus dados, através das tecnologias propostas pela Web Semântica, também tem se tornado uma necessidade, visando possibilitar a descoberta deste conteúdo por agentes inteligentes, que possibilitem algum nível de automação de tarefas e de reuso dos dados. A Web Semântica provê uma infraestrutura que permite o compartilhamento e reuso dos dados publicados, entre aplicações, empresa e comunidade. Utilizando essa infraestrutura, as organizações que desejam publicar seus

> dados na Web serão capazes de publicar dados anotados com um vocabulário específico de seus respectivos domínios [Salas et al ...]. A forma como as aplicações Web são projetadas atualmente, não contempla a possibilidade de tornar os dados disponíveis para outras aplicações os reutilizarem de alguma forma.

Portanto, os problemas que motivam este trabalho advêm da dificuldade de especificar o conteúdo dinâmico de aplicações Web de integração de dados, de forma que tanto a obtenção deste conteúdo quanto sua manutenção e uso sejam realizados de forma mais simples.

1.2 Data-Intensive Web Aplications

O termo Data-Intensive Web Application (DIWA) refere-se às aplicações Web, cujo principal objetivo é possibilitar o acesso e a manutenção de grandes quantidades de dados estruturados, normalmente armazenados em um sistema de gerenciamento de banco de dados [Ceri 2003]. Atualmente, este é o tipo de aplicação predominante na Web; sites de comércio eletrônico, portais corporativos e sites de comunidades são exemplos de aplicações DIWA.

Em aplicações DIWA, o conteúdo de cada página é especificado em termos de unidades de conteúdo, de forma que uma página pode ser vista como um *container* de diversas unidades de conteúdo. O termo Unidade de Conteúdo Dinâmico (UCD) é utilizado para designar unidades de conteúdo cujos dados são dinamicamente extraídos de uma ou mais fontes de dados [Ceri 2003, Manolescu 2005].

Quanto à implementação, em geral, estas aplicações são baseadas em uma arquitetura denominada multicamadas ou de três camadas [Fraternali 1999], a qual separa a lógica da aplicação, do conjunto de dados; introduzindo uma distinção de responsabilidades no lado do servidor, enquanto garante o mínimo de esforço no lado cliente. Nesta arquitetura, mostrada na Figura 1.1, a lógica de negócios é desempenhada pelo lado servidor, durante o atendimento às requisições do navegador do lado cliente.

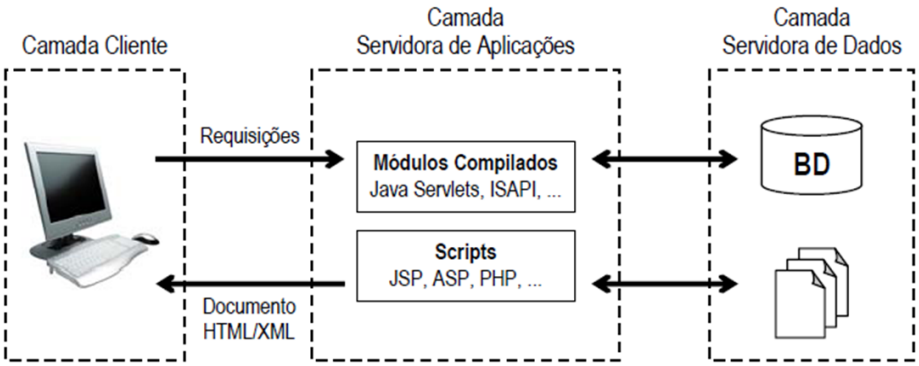


Figura 1.1 – Arquitetura de 3 camadas de aplicações DIWA

As três camadas que compõem esta arquitetura são as seguintes: (i) Camada Servidora de Dados, na qual os dados são persistidos e podem ser organizados e selecionados; (ii) Camada Servidora de Aplicações, que comporta os módulos responsáveis pela execução da lógica de negócios da aplicação, os quais obtém os dados requeridos pelas visões dos usuários; e (iii) Camada Cliente, que apresenta o conteúdo das páginas Web, geradas pela Camada Servidora de Aplicações.

Na arquitetura de uma aplicação DIWA, um tópico importante é a forma como o conteúdo da fonte de dados é apresentado nas páginas da aplicação. Esta apresentação pode ser feita de forma estática, quando páginas são computadas na definição da aplicação e permanecem imutáveis, durante o uso da aplicação; ou de forma dinâmica, quando o conteúdo das páginas é criado de forma *just-in-time*, isto é, à medida que o usuário interage com a aplicação para definir as informações que deseja obter.

1.3 Aplicações Mashup

Mashup é uma estratégia para o desenvolvimento de aplicações, que permite que os usuários agreguem múltiplas funcionalidades, cada uma com um propósito específico, para criar uma nova funcionalidade, com um propósito novo, bem definido. Conceitualmente, um Mashup é uma aplicação Web que combina informações e funcionalidades de múltiplas fontes na Web. [Lorenzo et al. 2009].

Alguns exemplos de Mashups disponíveis na Web hoje são: ChicagoCrime.org, o qual reúne dados de crimes da base de dados do Departamento Policial de Chicago com dados cartográficos do Google Maps; e o Diggdot.us, que combina mensagens (*feeds*) de fontes voltadas para tecnologias como Digg.com, Slashdot.org e Del.icio.us.

Segundo [Maximilien et al. 2007], uma aplicação Mashup possui os seguintes componentes: (i) Nível de Dados (ou definição do mashup), concernente à mediação e integração de dados; (ii) Nível de Processos (execução do mashup), onde é definida a composição entre as aplicações envolvidas; e (iii) Nível de Apresentação (interface do mashup), que constitui a interface com o usuário, utilizada para receber dados do usuário e para exibir informações relativas ao processo executado. A figura 1.2 apresenta um esquema de arquitetura de mashup, seguindo os níveis descritos.

Dentre estes níveis, nos interessa particularmente o nível de dados, isto é, a definição do mashup. As necessidades encontradas neste nível envolvem o acesso e a integração dos dados que residem em múltiplas fontes heterogêneas. Em geral, estes recursos são acessíveis através de serviços Web SOAP ou REST, protocolo HTTP e XML RPC. Com relação à mediação dos dados, os problemas básicos são relativos às diversidades estruturais e semânticas entre os esquemas a serem integrados. Além disso, as fontes das quais os dados são extraídos podem ser estruturadas, i.e., com um modelo de dados definido, como é o caso de documentos XML e de mensagens RSS/ATOM; ou não estruturadas, e.g., texto de email, arquivos de áudio e documentos. Em caso de dados não estruturados, estes devem ser pré-processados, de forma que um significado dados estruturados sejam criados a partir destes. Percebe-se que este nível consiste das diversas manipulações de dados necessárias para a integração de diferentes fontes de dados (conversão filtragem, transformação de formato etc.), onde capa manipulação deve considerar requisitos sintáticos e semânticos.

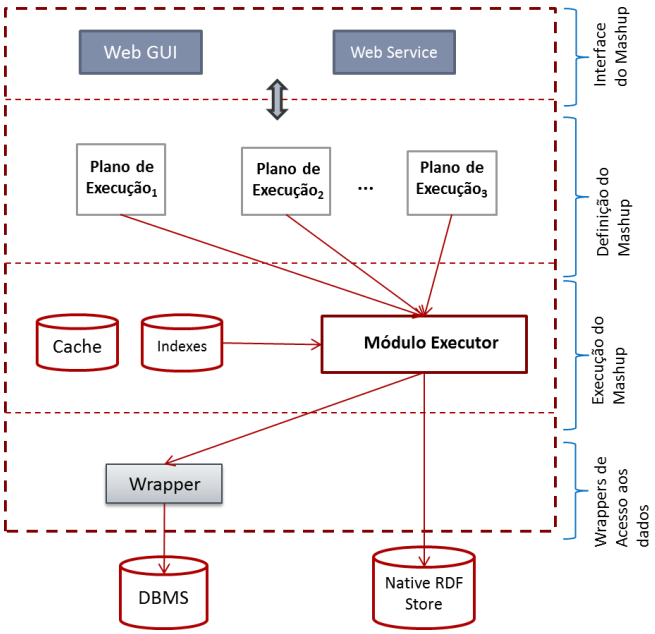


Figura 1.2 – Arquitetura de mashup

Os dados manipulados por um Mashup podem ser obtidos de duas formas [Merril 2006]: (i) provedores de conteúdo fornecem dados voluntariamente, através da disponibilização de APIs (interfaces de programação de aplicativos) que permitem o acesso e a manipulação dos dados da aplicação; ou (ii) através da técnica de extração de dados *screen scraping,* que utiliza ferramentas de software para analisar o conteúdo das páginas, que foram originalmente escritos para consumo humano, para extrair estruturas de dados semânticas, representativas dessa informação, que possam ser utilizados e manipulados através de programação.

Atualmente, não há uma padronização para a construção de Mashups, de forma que cada empresa, tais quais: Google, Microsoft e Yahoo, por exemplo, disponibilizam sua própria API, através da qual seus dados podem ser acessados. No entanto, muitas fontes de dados potencialmente interessantes (ainda) não disponibilizam convenientemente suas APIs, obrigando os desenvolvedores de Mashups a recorrerem à técnica de *screen scraping* de extração de dados para obterem as informações que pretendem combinar.

Existem atualmente diversas ferramentas para suporte à criação de Mashups, entretanto, na maioria dos casos o desenvolvimento de uma nova aplicação é realizado manualmente. Isso ocorre devido às limitações destas ferramentas e à dificuldade de reutilização de Mashups desenvolvidos com ferramentas específicas [Yu et al 2008]. Existem ainda abordagens que propõem uma arquitetura de integração de dados para Mashups, onde são utilizadas definições baseadas em *scripts* para facilitar a obtenção dinâmica do conteúdo para os Mashups [Thor et al 2009]. Entretanto, esta solução também não resolve a dificuldade na especificação do conteúdo, uma vez que é necessário o conhecimento e uso de uma linguagem de *scripts* específica, além de que a geração destes *scripts* é realizada manualmente.

1.4 Descrição da Proposta e Contribuições

A proposta deste trabalho é apresentar uma abordagem que visa minimizar os três principais problemas citados anteriormente, quanto ao desenvolvimento e utilização de Aplicações Web de Integração de Dados: (i) falta de um ciclo de desenvolvimento bem definido; (ii) dificuldade na especificação do conteúdo dinâmico e (iii) dificuldade na reutilização dos dados obtidos. Nossa principal contribuição consiste na definição de um método (Figura 1.3) que visa minimizar o problema da falta de um ciclo de desenvolvimento e que abrange os seguintes aspectos, os quais serão posteriormente detalhados neste trabalho:

(i) *a especificação conceitual, em alto nível, dos requisitos de conteúdo de aplicações Web de integração de dados* - Esta especificação é realizada sobre uma ontologia que representa um esquema conceitual unificado da aplicação Web (Ontologia da Aplicação). Dessa forma, a especificação do conteúdo dinâmico torna-se mais simples, visto que não há necessidade de conhecer os esquemas das diversas fontes a serem acessadas, nem de se utilizar diversas APIs e linguagens de programação. Além disso, as relações semânticas entre os termos das fontes são definidas através de mapeamentos entre a Ontologia da Aplicação e os esquemas das fontes (através de uma arquitetura de integração semântica);

(ii) *A geração automática dos planos de execução que calculam dinamicamente o conteúdo solicitado*. A partir de uma consulta definida sobre a Ontologia da Aplicação, é gerado um plano de execução contendo a sequência otimizada de operações para integrar dados. Tais planos são executados sobre as diversas fontes de dados que a aplicação necessita, as quais deverão estar semanticamente integradas, eliminando o problema de especificação manual das consultas de integração de dados sobre as várias fontes.

(iii*) O uso de anotações baseadas em ontologias*, para permitir a descoberta e reutilização deste conteúdo por agentes inteligentes e outras aplicações de integração de dados, desta forma, minimizamos o problema de dificuldade na reutilização dos dados obtidos.

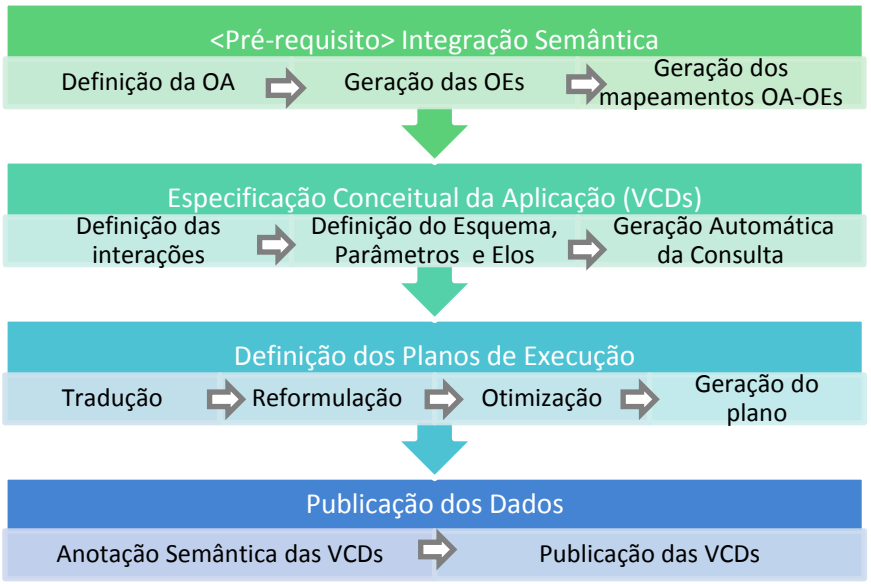


Figura 1.3 – Etapas do Método para projeto de aplicações Web de integração de dados

A hipótese geral deste trabalho é que um método, baseado nestes três componentes, traz melhorias significativas no desenvolvimento e na manutenção de aplicações Web de integração de dados. A principal vantagem da utilização deste método, além da padronização do projeto deste tipo de aplicação, está na possibilidade de especificar o conteúdo dinâmico em alto nível e então obter automaticamente a consulta e o plano de execução que extrai os dados das fontes, além de prever a reutilização semântica dos dados. Os passos do método serão posteriormente discutidos e detalhados, nos demais capítulos desta dissertação.

Visando obter os elementos necessários para a solução proposta, as contribuições deste trabalho são:

1. Definição e descrição de um método para o projeto de aplicações Web de integração de dados, que envolve três tarefas principais: especificação dos requisitos da aplicação sobre uma ontologia que a descreve (denominada Ontologia de Aplicação), onde tais requisitos são a especificação de quais dados devem ser apresentados como resultado; definição do plano de execução que calcula dinamicamente o conteúdo da aplicação Web; e anotação semântica do conteúdo obtido, seguida de sua publicação.

2. Dentre as etapas definidas no método, nossos esforços se concentram na etapa de Especificação Conceitual da Aplicação, sendo a definição da mesma contribuição nossa. Desta forma, apresentamos a definição e descrição de uma infraestrutura baseada em ontologias para permitir a especificação conceitual do conteúdo da aplicação Web, em termos de visões virtuais parametrizadas, denominadas Visões de Conteúdo Dinâmico (VCDs). Quanto às demais etapas do método (Definição dos Planos e Publicação dos Dados), citaremos outros trabalhos que podem eficientemente aplicados ou adaptados para a execução destas etapas;

3. Definição de um ambiente composto por um conjunto de módulos, que ofereça suporte a cada passo do método proposto;

4. Projeto e prototipação de uma ferramenta que permita o gerenciamento de VCDs para uma aplicação Web (componentes da Infraestrutura de Integração de Dados, e.g., criação, edição e remoção).

1.5 Trabalhos Relacionados

Relativos à Especificação de aplicações DIWA. Muitas plataformas e linguagens têm sido desenvolvidas para descrever aplicações DIWA, dentre as quais podemos citar OOHDM [Schwabe et al. 2002] e WebML [Manolescu et al. 2005].

Object-Oriented Hypermedia Design Method, é um método para desenvolvimento de aplicações Web que procura resolver as dificuldades encontradas na construção de projetos dessas aplicações, aplicando as estratégias da Engenharia de Software. A metodologia OOHDM apresenta uma abordagem orientada a objetos, que utiliza cenários e casos de uso para especificação das tarefas a serem realizadas pela aplicação. Diagramas de Interação de Usuários (UIDs) são utilizados para representação das interações entre os usuários e o sistema para a execução de cada tarefa.

Em [Manolescu 2005] é apresentado um processo de desenvolvimento de aplicações Web, composto das seguintes fases: Especificação de Requisitos, Projeto de Dados, Projeto de Hipertexto e Desenvolvimento da Aplicação. A fase de Desenvolvimento da Aplicação consiste nas seguintes atividades: Projeto de Arquitetura, Implementação dos Dados, Implementação do Hipertexto e Testes e Manutenção. Para especificação do hipertexto de aplicações DIWA, é proposta uma linguagem, chamada WebML (Web Modeling Language). WebML possui conceitos visuais para expressar o hipertexto como um conjunto de páginas, compostas por unidades de conteúdo ligadas entre si, operações para especificar as correspondências entre unidades e operações com os dados as quais estas se referem.

Para apoiar as diversas fases que compõem o processo WebML, foi desenvolvida a ferramenta WebRatio BPM [Brambilla 2010]. Esta ferramenta provê interfaces gráficas para que o usuário possa definir o esquema entidade-relacionamento e o esquema de hipertexto da aplicação. Além disso, WebRatio apoia a fase de implementação, automatizando a construção de um banco de dados relacional e dos *templates* das páginas, contendo as consultas SQL que extraem os dados de cada página. Essas consultas são geradas automaticamente, a partir da definição das páginas, durante o projeto de hipertexto e se restringem a consultas definidas sobre bancos de dados relacionais e são consultas simples, definidas sobre uma única tabela.

Relativos à Implementação Automática de UCDs. Dentre trabalhos que disponibilizam mecanismos para implementação automática do conteúdo dinâmico de aplicações DIWA, encontram-se ProDIWA, DataX e o trabalho de [Lemos 2007]. Detalharemos estes trabalhos a seguir.

Em [Lemos 2007], os requisitos de conteúdo dinâmico de cada página de uma aplicação Web são especificados através de uma visão XML, denominada Visão de Navegação (VN). É proposto um processo para especificação e geração de Visões de Navegação para aplicações Web, cujo conteúdo é extraído de uma ou mais fontes de dados. São consideradas fontes de dados relacionais e XML. As visões de navegação são implementadas como visões XML, as quais são especificadas conceitualmente usando o formalismo das assertivas de correspondência propostas no mesmo trabalho. O formalismo proposto para a especificação das VNs é independente de linguagem de consulta e permite que as definições das VNs sejam geradas automaticamente.

A proposta de DataX [Vidal et al. 2008] é um framework para a geração de serviços web AXML para materialização do conteúdo dinâmico das páginas web em aplicações DIWA. Neste framework, o conteúdo dinâmico de uma página web é definido como uma visão virtual, que por sua vez, é especificada com a ajuda de um conjunto de assertivas de correspondência, que especificam mapeamentos entre o esquema da visão XML e os esquemas das fontes locais. Com base nas assertivas de correspondência da visão, o serviço web que materializa o conteúdo da visão é gerado automaticamente.

Em ProDIWA [Vidal et al. 2006], os requisitos de conteúdo de cada página da aplicação são especificados através de uma visão, denominada Visão de Contexto de Navegação (VCN). O trabalho considera que os dados das VCNs estão armazenados em um banco de dados relacional, e que o mesmo pode já existir, ou que deverá ser projetado especificamente para a aplicação. Neste trabalho, propõe-se um processo para geração e manutenção das VCNs para aplicações DIWA. Uma das vantagens do enfoque proposto é que a implementação e a manutenção das VCNs podem ser realizadas de forma automática a partir de suas especificações conceituais. As VCNs podem ser implementadas como visões de objeto ou como visões XML. Entretanto, a recuperação do conteúdo é feita sobre uma única base.

Embora lidem com a geração automática do conteúdo dinâmico das páginas de aplicações DIWA, estes trabalhos apresentam algumas faltas. O complexo problema da geração e manutenção dos mapeamentos definidos pelas assertivas de correspondência não está no escopo destes trabalhos. A geração e manutenção destes mapeamentos são naturalmente complexas, uma vez que os mapeamentos são entre estruturas XML. Além disso, nenhum destes trabalhos considera o problema da integração semântica dos dados, portanto, não resolvem problemas de interoperabilidade.

1.6 Organização da Dissertação

Os capítulos a seguir estão organizados da seguinte forma. No Capítulo 2, apresentamos os conceitos de Integração Semântica, abordando algumas arquiteturas que podem ser utilizadas e apresentamos uma Infraestrutura básica da Integração de Dados. No Capítulo 3 introduzimos e discutimos uma visão mais detalhada do método proposto. No Capítulo 4, abordamos mais especificamente os passos da etapa de Especificação Conceitual e de Definição dos Planos da Aplicação, além de mostrar um estudo de caso para exemplificar os passos destas etapas. No Capítulo 5, apresentamos o conceito de descrições semânticas dos dados, bem como sua importância e discutimos a etapa de Publicação de Dados. No Capítulo 6, descrevemos ambiente conceitual que possibilita a implantação do método apresentado, detalhando cada um dos módulos e artefatos pertinentes a este, além de apresentar o protótipo da ferramenta concebida para instanciar o ambiente proposto. Por fim, no Capítulo 7 são mostradas as conclusões e sugestões de trabalhos futuros.

2. Integração Semântica

Este capítulo apresenta os conceitos de Integração Semântica, abordando alguns exemplos arquiteturas que podem ser utilizadas para a criação de um ambiente de integração semântica. Na seção 2.1 apresentamos a definição de Integração de Dados e Integração Semântica. Na seção 2.2 discutimos dois tipos de arquiteturas utilizadas para Integração Semântica: arquitetura de dois níveis de ontologias e Linked Data. Na seção 2.3 explicamos como uma ontologia pode ser utilizada para representar o esquema conceitual de uma aplicação Web. Após isto, na seção 2.4 explicamos os enfoques para representação de fontes de dados através de ontologias. Por fim, na seção 2.5 apresentamos uma infraestrutura baseada em ontologias, utilizada como base para a especificação de Aplicações Web de Integração de Dados

2.1 Integração de Dados

Integração de Dados é definida na literatura como o problema de combinar dados, que residem em diferentes fontes, e prover o usuário de uma visão unificada destes dados [Lenzerini 2002]. Em um típico sistema de integração de dados, observa-se uma arquitetura baseada em um esquema global e um conjunto de fontes. As fontes contêm os dados reais, enquanto o esquema global provê uma visão virtual e integrada dos dados contidos nas fontes. Um conjunto de mapeamentos, entre o esquema global e os esquemas das fontes, estabelece as correspondências entre estes. Dessa forma, o sistema de integração deve ser capaz de responder a consultas submetidas ao sistema, em termos do esquema global.

Frequentemente, as fontes de dados a serem integradas apresentam esquemas e modelos distintos, além de se encontrarem geograficamente distribuídas. Estas características nos remetem ao problema de reunir sistemas computacionais heterogêneos e distribuídos, conhecido como problema de interoperabilidade [Wache et al. 2001]. Os problemas de interoperabilidade que podem surgir, em decorrência da heterogeneidade dos dados, são bastante conhecidos na comunidade de banco dados: heterogeneidade estrutural (de esquemas) e heterogeneidade semântica (de dados) [Kim and Seo 1991]. A heterogeneidade estrutural é decorrente do fato de que sistemas de informação distintos armazenam seus dados em estruturas distintas. Por sua vez, a heterogeneidade semântica se refere ao conteúdo da informação e seu significado. Visando atingir a interoperabilidade semântica, em um sistema de integração de dados, o significado das informações deve ser compartilhado pelos componentes do sistema.

Desta forma, é possível referir-se à Integração Semântica como o problema de combinar dados, que residem em diferentes fontes, e prover o usuário com uma visão semanticamente unificada destes dados. Isto é, de forma que as diferenças semânticas entre as fontes de dados e o esquema global sejam reconciliadas. O uso de ontologias para a explicação do conhecimento implícito, dos dados integrados, é uma abordagem possível para o problema de heterogeneidade semântica. [Wache e al. 2001].

Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada [Gruber 1995], onde uma conceitualização pode ser definida como a forma de pensar sobre determinado domínio [Uschold 1998]. O uso de ontologias para descrição do conjunto de esquemas-fonte, bem como para representação do esquema global de um sistema de integração de dados, enriquece este sistema, ajudando a transpor o problema de heterogeneidade semântica. Segundo [Hakimpour and Geppert 2002], a transposição deste problema, através do uso de ontologias, é possível devido ao fato de que uma ontologia formal consiste em axiomas lógicos, que transmitem o significado dos termos em uma determinada comunidade. Por sua vez, um conjunto de axiomas lógicos para a definição de um termo, é denominado definição intencional e há somente uma definição intencional para um termo, dentro de uma comunidade, o que permite a eliminação das ambiguidades de significado, trazidas pela heterogeneidade semântica.

2.2 Arquiteturas de Integração Semântica

Neste trabalho, a integração semântica das fontes de dados é uma peça fundamental, pois o método que apresentamos necessita de uma estrutura de integração semântica para viabilizar a especificação conceitual em alto nível, do conteúdo de aplicações Web de integração de dados. Esta especificação é realizada sobre uma ontologia que representa um esquema conceitual unificado da aplicação Web (Ontologia da Aplicação). Dessa forma, a especificação do conteúdo dinâmico torna-se mais simples, visto que não há necessidade de conhecer os esquemas das diversas fontes a serem acessadas, nem de se utilizar diversas APIs e linguagens de programação. Salientamos que quaisquer arquiteturas que proporcionem a integração semântica utilizando algum tipo de esquema conceitual unificado, poderão ser utilizadas pelo método apresentado neste trabalho. A seguir, apresentamos duas arquiteturas de integração semântica que têm sido atualmente difundidas na comunidade: arquitetura de dois níveis de ontologias [Calvanese et al. 2007], [Klien et al. 2007] e Linked Data [Hartig et al. 2009].

2.2.1 Arquitetura de Dois Níveis de Ontologias

A arquitetura de dois níveis de ontologias para a integração de dados (Figura 2.1) é composta pelos seguintes elementos: a Ontologia de Domínio (OD), a qual contém os termos básicos do domínio e representa o vocabulário compartilhado da aplicação; um conjunto de Ontologias Locais (OL), as quais descrevem as fontes de dados, utilizando uma linguagem de ontologia; e o mapeamento que especifica as correspondências entre as Ontologias Locais e a Ontologia de Domínio (mapeamentos OL-OD). Os sistemas propostos em [Calvanese et al. 2007] e [Klien et al. 2007], adotam esta arquitetura.

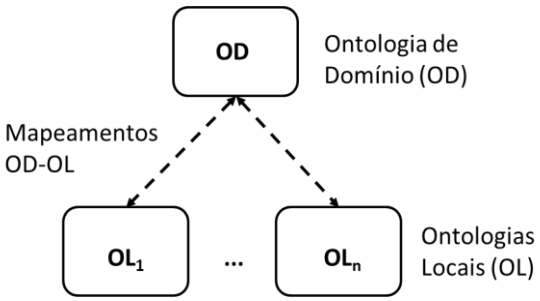


Figura 2.1 Arquitetura de dois níveis

Esta arquitetura é baseada no *matching* (i.e. operação que permite a comparação de similaridade) entre os conceitos presentes nas ontologias locais e na ontologia de domínio, gerando mapeamentos entre os conceitos das ontologias. Tais mapeamentos são utilizados para o desdobramento de uma consulta, submetida sobre a ontologia de domínio, em subconsultas expressas em termos das ontologias locais.

Segundo [Sacramento et al. 2010], os principais problemas relativos a esta arquitetura são: (i) como especificar os mapeamentos entre as ontologias e (ii) como utilizar os mapeamentos obtidos, para responder corretamente às consultas submetidas sobre a Ontologia de Domínio, a qual é utilizada para especificar o esquema de mediação.

2.2.2 Linked Data

Outra forma que tem sido proposta para a integração semântica de dados, especificamente na Web, é através do uso de Linked Data. Linked Data é um conjunto de melhores práticas para publicação e consumo de dados estruturados na Web, permitindo estabelecer links entre itens de diferentes fontes de dados para formar um único espaço de dados global [Hartig et al. 2009]. Através do uso de URIs, Linked Data manipula os recursos da Web como objetos individuais para serem formados e apresentados como conhecimento ou para agir como novos serviços e produtos [Hiatt, 2009]. Assim, Linked Data tem o potencial de facilitar o acesso aos dados semanticamente relacionados, estabelecendo conexões explícitas entre conjuntos de dados (datasets), como pode ser visto na Figura 2.2.

Devido a esta conexão, aplicações Web que visam integrar dados podem se beneficiar do ambiente Linked Data, criando um esquema conceitual integrado (Ontologia da Aplicação) mapeado sobre as ontologias existentes no Linked Data, obtendo assim as vantagens de reutilizar vocabulários já bem estabelecidos na comunidade.

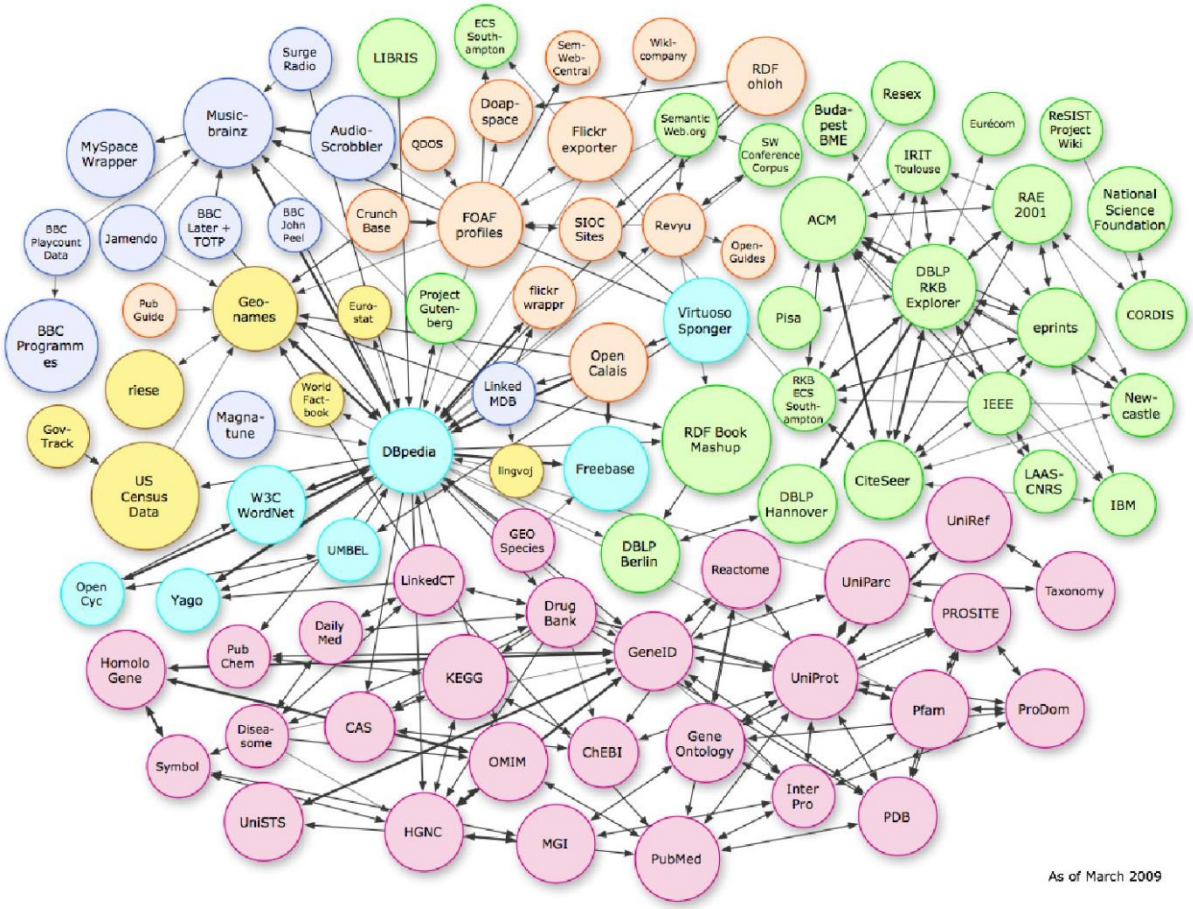


Figura 2.2 Datasets no Linked Data (linkeddata.org, março de 2009)

A Web de dados está baseada em alguns padrões: um mecanismo de identificação global e único (*URIs – Uniform Resource Identifiers*), um modelo de dados comum (*RDF - Resource Description Framework*) e uma linguagem de consulta para acessar os dados (SPARQL). Esses padrões são descritos a seguir:

- URIs [Berners-Lee et al. 2005] são usadas no contexto de *Linked Data* para identificar objetos e conceitos, permitindo que eles sejam dereferenciados para obtenção de informações a seu respeito. Assim, uma URI dereferenciada resulta em uma descrição RDF do recurso identificado.

- A utilização um modelo de dados comum – modelo RDF – torna possível a implementação de aplicações genéricas capazes de operar sobre o espaço de dados global [Bizer et al. 2009]. O modelo RDF [Manola and Miller, 2004] é um modelo de dados descentralizado, baseado em grafo e extensível, possuindo um alto nível de expressividade e permitindo a interligação entre dados de diferentes fontes. Ele foi projetado para a representação integrada de

informações originárias de múltiplas fontes. Os dados são descritos na forma de triplas com sujeito, predicado e objeto. Cada tripla faz parte da Web de Dados e pode ser usada como ponto de partida para explorar esse espaço de dados. Triplas de diferentes fontes podem ser facilmente combinadas para formar um único grafo. Além disso, é possível usar termos de diferentes vocabulários para representar os dados. Links RDF podem estabelecer relações entre dados de diferentes conjuntos de dados.

- Consultas à Web de Dados podem ser realizadas através da linguagem SPARQL [PRUD'HOMMEAUX, E. and Seaborne, 2008], que é a linguagem de consulta padrão da Web Semântica para recuperação de informações contidas em grafos RDF. No entanto, SPARQL não é somente uma linguagem de consulta declarativa, mas também um protocolo [Clark et al. 2008] usado para enviar consultas e recuperar resultados através do protocolo HTTP. Fontes *Linked Data* tipicamente fornecem um *SPARQL endpoint* que é um serviço Web com suporte ao protocolo SPARQL. Esse serviço possui uma URI específica para receber requisições HTTP com consultas SPARQL e retornar os resultados dessas consultas.

Além do uso dos padrões acima descritos, a Web de dados possui as seguintes características de acordo com [Bizer et al 2007]: É genérica e pode conter qualquer tipo de dado; qualquer pessoa pode publicar dados; não há restrições para seleção de vocabulários; os dados são auto descritos, de modo que ao dereferenciá-los é possível obter sua definição; o uso de um mecanismo padrão de acesso aos dados (HTTP) e de um modelo de dados padrão (RDF) simplifica o acesso aos dados; as aplicações que usam a Web de dados não ficam limitadas a um conjunto fixo de fontes de dados, podendo inclusive descobrir novas fontes de dados em tempo de execução ao seguir *links RDF*.

2.3 Representação do Esquema Conceitual de Aplicações Web através de Ontologias

O esquema conceitual de uma aplicação Web pode ser visto como uma representação de conceitos presentes da aplicação e seus relacionamentos. O esquema conceitual descreve a semântica de uma aplicação e apresenta uma série de declarações sobre sua natureza. Especificamente, descreve as entidades significativas para a aplicação, sobre as quais coleta informações e características (atributos), bem como as associações entre pares de entidades (relacionamentos) e pode ser representado através de diversos modelos conceituais.

Segundo [Guizzardi et al. 2009], modelos conceituais são artefatos produzidos com o objetivo de representar uma determinada porção da realidade, segundo uma determinada conceituação; e existem tipos específicos de ontologias que descrevem as categorias utilizadas para a construção dessas conceituações. Dessa forma, ontologias podem ser utilizadas como um tipo particular de modelo conceitual, portanto podem também ser utilizadas para representar o esquema conceitual de uma aplicação Web.

O termo “ontologia” tem sido empregado na computação, no sentido de descrever um conjunto de definições de conceitos, propriedades, relações, restrições, axiomas, processos e eventos que descrevem certo domínio ou universo de discurso. Essas definições permitem que aplicações e agentes de software utilizem uma semântica formal, precisa e livre de ambiguidades, para processar a informação descrita pela ontologia e utilizar essa informação em aplicações inteligentes [Prazeres 2010]. Descrever o esquema conceitual de uma aplicação Web através de uma ontologia possibilita que, os recursos disponibilizados pela aplicação, sejam utilizados por agentes na automatização de tarefas diversas, o que insere a aplicação Web no contexto da Web Semântica.

Para descrever o esquema conceitual através de uma ontologia, é necessária uma correspondência entre os conceitos definidos no esquema conceitual, e os elementos que compõem uma ontologia. Desta forma, as entidades do esquema conceitual são representadas, na ontologia, por classes; os atributos das entidades são representados por propriedades de dados; os relacionamentos entre entidades são representados por propriedades de objetos; e as regras de negócio da aplicação, por axiomas definidos na linguagem de ontologia utilizada. Em nossa abordagem, a ontologia utilizada para descrever uma aplicação Web é denominada Ontologia de Aplicação (OA).

[Gomez-Perez et al. 2004] definem dois tipos de ontologias: ontologia de domínio (ou vertical) e ontologia de nível superior, independente de domínio, ou ontologia horizontal. Ontologias de domínio definem conceitos para um domínio específico, por exemplo, biologia, matemática, patologias e vinhos. Ontologias de nível superior são aquelas em que os conceitos independem de domínio, de forma que podem ser utilizadas em diversos domínios, por exemplo, a ontologia DOLCE[1]. A Ontologia de Aplicação utilizada neste trabalho enquadra-se na categoria de ontologia de domínio específico, uma vez que descreve os conceitos relativos a uma aplicação Web em particular.

A descrição semântica do domínio das aplicações na Web Semântica traz diversas vantagens, como facilidade na descoberta e reuso de conteúdo e automação de tarefas. Outra vantagem do uso de ontologias na descrição de aplicações web é a possibilidade de utilizar a ontologia para descrever serviços Web que pertencem ao mesmo domínio, facilitando sua descoberta, por agente de software, na Web. Tais vantagens têm feito com que cada vez mais projetistas escolham utilizar ontologias para descrever suas aplicações.

2.4 Representação de Fontes de Dados através de Ontologias

Ontologias podem ser utilizadas para a representação dos esquemas e/ou do conteúdo de uma fonte de dados, permitindo a sua publicação na infraestrutura fornecida pela Web Semântica. Utilizando essa infraestrutura, as organizações que desejam publicar os dados de suas fontes na Web, serão capazes de publicar dados anotados com um vocabulário específico de seus respectivos domínios [Salas et al.2010]. Isto é, uma fonte poderá publicar seus dados (através de uma ontologia) em termos de um vocabulário específico para o domínio da aplicação.

[1] www.loa.istc.cnr.it/DOLCE.html

A maior parte das fontes utilizadas atualmente, para o armazenamento de dados, utiliza o modelo relacional ou estruturas XML. Fontes relacionais são as mais largamente utilizadas, devido à sua simplicidade e ao formalismo matemático que endossa este modelo. Por outro lado, XML traz interoperabilidade em um nível sintático e rapidamente estabeleceu-se como um padrão para intercâmbio de dados, sendo esta sua função principal. Dessa forma, as abordagens e ferramentas atualmente disponíveis para a construção de ontologias a partir de fontes de dados se baseiam, sobretudo, no modelo relacional, embora algumas forneçam também suporte para dados XML [Ghawi and Cullot, 2009]. Portanto, consideraremos neste trabalho, o modelo relacional como o modelo utilizado pelas fontes participantes da Integração Semântica.

A publicação dos dados de uma fonte através de uma ontologia pode ser realizada através de dois enfoques:

1. **Enfoque virtual** – os dados que residem nas fontes são acessados durante o processamento da consulta, a qual é definida sobre a ontologia. Neste enfoque, não há qualquer tipo de replicação dos dados da fonte, no entanto, a ontologia representa uma visão virtual do esquema da fonte. Para possibilitar o acesso aos dados, uma consulta realizada em termos da ontologia é traduzida para uma consulta sobre o esquema da fonte, tendo como base um conjunto de mapeamentos entre ambas as estruturas. Segundo [Ghawi and Cullot, 2009], diversas linguagens têm sido propostas para expressar formalmente o mapeamento entre fonte de dados e ontologia. Dentre estas, D2RQ[2] tem se destacado e tem sido bastante difundida e utilizada. D2RQ é uma linguagem declarativa baseada em RDF, para descrever mapeamentos entre fontes relacionais e ontologias OWL/RDFS. O objetivo da plataforma D2RQ é utilizar estes mapeamentos, para prover uma visão RDF de uma fonte de dados não-RDF. No mapeamento realizado pela D2RQ, conceitos básicos são definidos utilizando mapas de classes, que associam os conceitos da ontologia aos conjuntos de dados da fonte. Um mapa de classe especifica como as propriedades de uma instância são identificadas, além disso, possui um conjunto de pontes de propriedade, que especifica como as propriedades de uma instância são criadas. Pontes de propriedade de objeto devem possuir uma primitiva, a qual indica a ponte de conceito relacionada, e uma primitiva de junção que indica como as tabelas relacionadas devem sofrer junção. Os

[2] www4.wiwiss.fu-berlin.de/bizer/d2rq

bancos de dados suportados pela D2RQ são: Oracle[3], MySQL[4], PostgresSQL[5], Microsoft SQL Server[6] e fontes de dados ODBC[7].

2. **Enfoque materializado** – os dados que residem nas fontes são replicados em uma nova estrutura (ontologia), de forma que podem ser acessados diretamente, sem a necessidade de tradução de consulta sobre a fonte. Neste caso, a ontologia publicada é instanciada com os dados da fonte. Segundo [Ghawi and Cullot, 2009], as abordagens para mapeamento de uma fonte de dados para uma ontologia pode ser classificada em duas categorias:

   i. *Abordagens que criam uma ontologia a partir da fonte de dados.* O objetivo destas abordagens é a criação de uma ontologia a partir de uma fonte de dados relacional e a migração do seu conteúdo para a ontologia gerada. Os mapeamentos são as correspondências entre cada componente da ontologia criada (conceitos, propriedades etc.) e o componente original na fonte de dados (tabela, coluna etc.). Nestas abordagens, o modelo da fonte de dados e a ontologia gerada são bastante similares e, portanto, os mapeamentos são relativamente simples e diretos. São trabalhos que adotam esta abordagem: [Stojanovi et al, 2002], DataGenie[8] e Relational.OWL [Laborda and Conrad, 2005].

   ii. *Abordagens que mapeiam a fonte de dados para uma ontologia existente.* O objetivo destas abordagens é estabelecer mapeamentos entre uma ontologia existente e uma fonte de dados e/ou popular a ontologia com o conteúdo da fonte de dados. Neste caso, os mapeamentos são normalmente mais complexos do que na categoria anterior, porque os critérios de modelagem utilizados no projeto da fonte de dados são diferentes dos critérios utilizados na modelagem da ontologia. KAON Reverse[9] e Vis-a-Vis [Konstantinou et al. 2006] são exemplos de ferramentas que executam a proposta desta abordagem.

[3] www.oracle.com
[4] www.mysql.com
[5] www.postgresql.org
[6] www.microsoft.com/brasil/sql/
[7] support.microsoft.com/kb/110093
[8] www.data-genie.com
[9] kaon.semanticweb.org/alphaworld/reverse/

Neste trabalho, adotamos o enfoque virtual de publicação de dados, devido às vantagens desta abordagem, e.g., o fato de que as informações recuperadas estão constantemente atualizadas. Segundo [Salgado and Loscio, 2001], a abordagem virtual é mais adequada para os casos em que as informações mudam rapidamente e para consultas que operam sobre uma grande quantidade de dados distribuídos, em um grande número de fontes de dados. Este cenário é semelhante ao cenário das aplicações Web de Integração de Dados, as quais são o foco deste trabalho, o que justifica a escolha desta abordagem.

2.5 Infraestrutura Baseada em Ontologias para Especificação de Aplicações Web de Integração de Dados

Uma vez que a integração semântica é um pré-requisito para a utilização do método proposto neste trabalho para o projeto de aplicações Web de integração de dados, apresentamos, com base nos conceitos discutidos, uma infraestrutura que contém os elementos básicos para permitir a especificação de uma aplicação Web sobre uma arquitetura de integração semântica. Como citado anteriormente, diversas arquiteturas de integração semântica de dados podem ser utilizadas para a obtenção da integração necessária, esta infraestrutura visa, portanto, definir quais são os elementos essenciais para que o método que utilizaremos possa aproveitar adequadamente a integração semântica das fontes.

Esta infraestrutura (Figura 2.3) é composta por três elementos: (i) cada fonte de dados é encapsulada por um *wrapper* que fornece uma Ontologia Exportada (OE), a qual descreve um subconjunto do esquema do banco de dados; (ii) o esquema conceitual da aplicação Web é representado por uma Ontologia de Aplicação (OA) e (iii) de posse das ontologias que descrevem as fontes e da ontologia que descreve a aplicação Web, é possível que as diferenças semânticas entre as fontes de dados e o esquema global sejam reconciliadas, através do estabelecimento de um conjunto de mapeamentos entre as ontologias.

Em nossa abordagem, visando tirar proveito das vantagens do processamento de consultas sobre uma arquitetura de integração de dados baseada em ontologias [Pinheiro et al. 2010], os requisitos de dados da aplicação Web são especificados através de uma visão virtual parametrizada denominada Visão de Conteúdo Dinâmico (VCD), a qual é definida conceitualmente através de uma consulta SPARQL sobre a Ontologia da Aplicação, fornecendo aos usuários uma visão unificada das fontes ou *datasets* a serem consultados. A forma como as VCDs são especificadas sobre a OA será discutida detalhadamente no Capítulo IV.

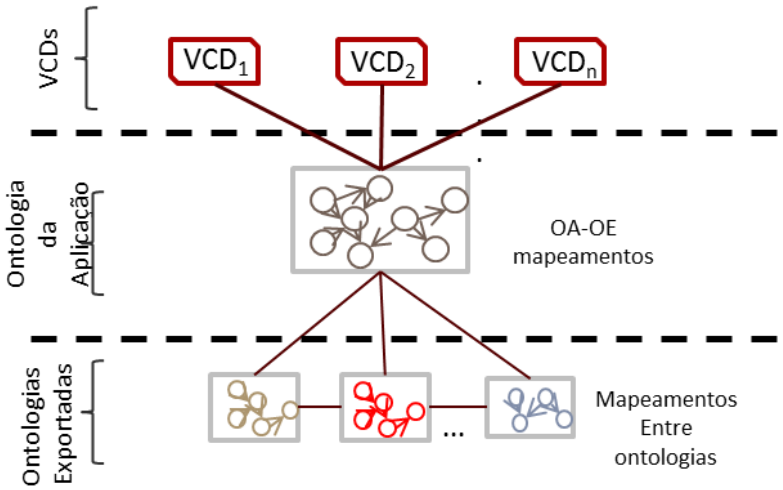


Figura 2.3 Infraestrutura Baseada em Ontologias para Especificação da Aplicação Web

No primeiro nível observam-se as VCDs da aplicação Web. Uma VCD é uma visão virtual parametrizada que representa os requisitos de conteúdo dinâmico de uma aplicação Web de integração de dados. A especificação das VCDs de uma aplicação é independente dos esquemas das fontes, uma vez que cada VCD da aplicação Web é definida conceitualmente através de uma consulta sobre a Ontologia da Aplicação, que representa o esquema conceitual da aplicação. Isto permite que o desenvolvedor da aplicação se abstraia dos detalhes relativos ao acesso a cada fonte de dados e possa projetar a aplicação de acordo com o esquema conceitual da aplicação.

O segundo nível da Infraestrutura de Especificação é composto por uma Ontologia da Aplicação (OA), a qual é uma ontologia de domínio específico que descreve conceitos de uma aplicação Web particular, ou seja, OA representa o esquema conceitual da aplicação. O terceiro nível é composto por um conjunto de Ontologias Exportadas (OE). Cada OE descreve um conjunto de dados de acordo com vocabulário da OA. Conceitos das OEs estão interligados através de mapeamentos interontológicos que estabelecem correspondências entre as OEs. Uma OE pode ser, por exemplo, uma ontologia publicada em Linked Data, mas também pode ser uma ontologia exportada diretamente de uma fonte de dados, assim como as Ontologias Locais da arquitetura de dois níveis. Há também um conjunto de mapeamentos entre OA e OEs que permite que uma consulta sobre a OA seja reescrita em subconsultas sobre as OEs. Utilizamos uma abordagem *Global-as-View* e cada VCD é definida sobre a visão global, i.e., a Ontologia da Aplicação, de forma que as especificações de VCDs são independentes das fontes de dados consultadas, uma vez que são definidas sobre uma visão unificada das mesmas.

2.6 Considerações Finais

Neste capítulo, tratamos sobre a Integração Semântica, i.e, o problema de combinar dados, que residem em diferentes fontes, e prover o usuário com uma visão semanticamente unificada destes dados. Este é um problema atual e recorrente, devido à grande quantidade de dados distribuídos, a qual as aplicações Web necessitam acessar. Visando prover o usuário de uma aplicação Web com uma visão semanticamente integrada dos dados, baseamos este trabalho numa infraestrutura que pode ser utilizada com diversas arquiteturas de integração semântica. Dessa forma, apresentamos as abordagens e ferramentas que podem ser utilizadas para a geração de cada uma das ontologias utilizadas e dos mapeamentos necessários entre estas.

3. Método para Projeto de Aplicações Web

Neste capítulo apresentamos nossa proposta, um método para definir o projeto de aplicações Web de integração de dados. Na Seção 3.1 oferecemos uma visão geral deste método explicando seus objetivos. Na Seção 3.2 apresentamos e discutimos brevemente os passos das etapas que compõem o método. Por fim, na Seção 3.3 apresentamos os passos da etapa de Integração Semântica, que constitui um pré-requisito para as demais etapas do método.

3.1 Visão Geral

No Capítulo I, discutimos a motivação e a necessidade de propor novas abordagens, para solucionar as dificuldades na especificação das unidades de conteúdo dinâmico e na automatização de sua implementação. Conteúdo dinâmico, neste tipo de aplicação, é visto como o conteúdo solicitado sob demanda pelo usuário, de acordo com um conjunto de parâmetros determinados pelo mesmo, durante sua interação com a aplicação. Uma das principais dificuldades apresentadas é a falta de um ciclo de desenvolvimento bem definido para este tipo de aplicação. Entretanto, visto que o uso de uma metodologia é capaz de trazer diversas vantagens para este processo, tais quais: aumento na qualidade, independência de indivíduos, facilidade de manutenção, aumento da reutilização de módulos entre outras, o principal objetivo deste trabalho consiste na definição de um método para o projeto de aplicações Web de integração de dados. Este método abrange os seguintes aspectos:

(i) *a especificação conceitual, em alto nível, dos requisitos de conteúdo de aplicações Web de integração de dados* - Esta especificação é realizada sobre uma ontologia que representa um esquema conceitual unificado da aplicação Web (Ontologia da Aplicação). Dessa forma, a especificação do conteúdo dinâmico torna-se mais simples, visto que não há necessidade de conhecer os esquemas das diversas fontes a serem acessadas, nem de se utilizar diversas APIs e linguagens de programação. Este método segue uma abordagem centrada no usuário, uma vez que depende da coleta de requisitos do mesmo para a determinação dos atributos e parâmetros das unidades de conteúdo dinâmico. Além disso, as relações semânticas entre os termos das fontes são definidas através de mapeamentos entre a Ontologia da Aplicação e os esquemas das fontes (através de uma arquitetura de integração semântica), o que implica que nosso método possui como pré-requisito a realização de uma integração semântica entre as fontes que serão acessadas pela aplicação;

(ii) *A geração automática dos planos de execução que calculam dinamicamente o conteúdo solicitado*. A partir de uma consulta definida sobre a Ontologia da Aplicação, é gerado um plano de execução contendo a sequência otimizada de operações para integrar dados. Tais planos são executados sobre as diversas fontes de dados que a aplicação necessita, as quais deverão estar semanticamente integradas, eliminando o problema de especificação manual das consultas de integração de dados sobre as várias fontes.

(iii*) O uso de anotações baseadas em ontologias*, para permitir a descoberta e reutilização deste conteúdo por agentes inteligentes e outras aplicações de integração de dados, desta forma, minimizamos o problema de dificuldade na reutilização dos dados obtidos.

3.2 Etapas do Método

O método que apresentamos neste trabalho (Figura 1.3) permite a implementação automática das unidades de conteúdo dinâmico de aplicações Web de integração de dados, a partir de sua especificação conceitual em alto nível, e tem como pré-requisito o estabelecimento da integração semântica entre as fontes envolvidas. Dessa forma, o método é baseado em três tarefas principais: (i) especificação dos requisitos da aplicação sobre uma ontologia que a descreve (denominada Ontologia de Aplicação), onde tais requisitos são a especificação de quais dados devem ser apresentados como resultado. Tais requisitos são especificados através de uma visão parametrizada denominada Visão de Conteúdo Dinâmico (VCD), a qual é definida conceitualmente através de uma consulta sobre a OA (ii) definição do plano de execução que calcula dinamicamente o conteúdo da aplicação Web; e (iii) anotação semântica do conteúdo obtido, seguida de sua publicação.

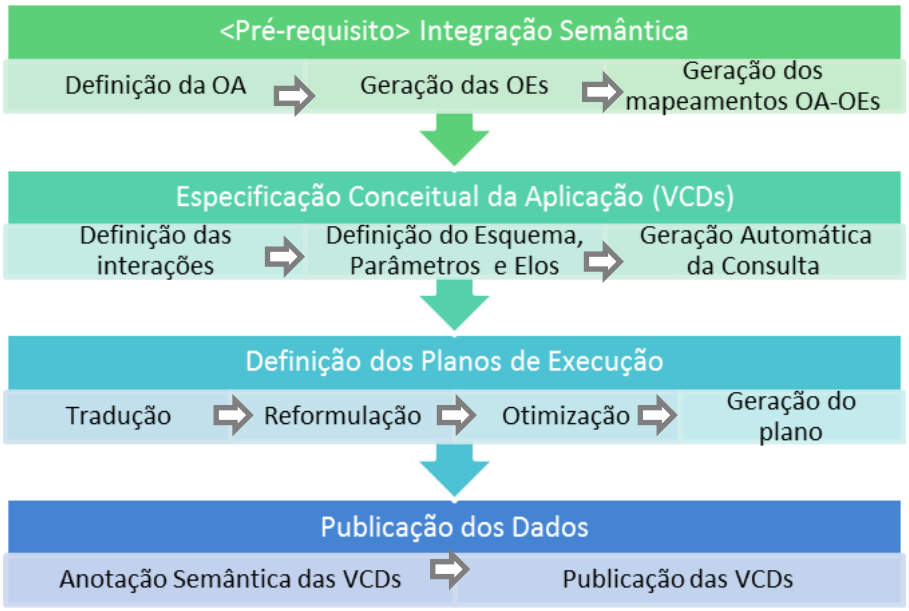


Figura 1.3 – Etapas do Método para projeto de aplicações Web de integração de dados

A etapa de Integração Semântica é um pré-requisito para a execução do método e consiste em prover ao usuário da aplicação Web uma visão unificada dos dados, contidos nas diversas fontes acessadas pela aplicação. Esta etapa pode ser omitida caso as fontes que serão consultadas pela aplicação já estejam semanticamente integradas. Para a realização da Integração Semântica, o esquema conceitual da aplicação Web é definido por uma Ontologia de Aplicação (OA); as fontes locais a serem integradas publicam visões RDF de seus dados, em termos do vocabulário da OA, constituindo as Ontologias Exportadas (OE); por fim, um conjunto de mapeamentos especifica as correspondências entre OA e OEs. Tais passos serão detalhados adiante na Seção 3.2.

A etapa de Especificação Conceitual da Aplicação consiste em determinar os requisitos de conteúdo das unidades de conteúdo dinâmico, em termos de visões virtuais parametrizadas, definidas como uma consulta SPARQL sobre a Ontologia da Aplicação, denominadas VCDs. Para cada interação necessária entre o usuário e o sistema, para a realização de uma tarefa que demande recuperação de conteúdo dinamicamente, há uma VCD correspondente. Sendo esta etapa do método nosso foco neste trabalho, os passos da Especificação Conceitual serão considerados mais detalhadamente no Capítulo IV, juntamente com um estudo de caso.

A etapa de Definição dos Planos de Execução das VCDs consiste na geração do plano de execução da consulta que define a VCD e na disponibilização deste plano através de um mecanismo que permita o acesso ao usuário. Propomos o uso de serviços Web como mecanismo de acesso responsável por materializar o conteúdo das VCDs da aplicação. O plano de execução da consulta de cada VCD é gerado automaticamente, a partir da consulta Q, que define a VCD sobre a OA e dos mapeamentos entre a OA e as OEs. Os passos da etapa de Implementação da VCD serão tratados no Capítulo IV, onde sugerimos ainda alguns trabalhos cujos resultados podem ser utilizados para a execução desta etapa.

A etapa de Publicação de VCDs consiste em possibilitar que as unidades de conteúdo dinâmico de uma aplicação sejam semanticamente anotadas e disponibilizadas em catálogos na Web, possibilitando sua reutilização por outras aplicações. Trataremos sobre esta etapa no Capítulo V, onde discutimos sobre anotação semântica e expomos uma forma sobre como este conceito pode ser aplicado à nossa proposta, além de discutirmos sua publicação e encontrabilidade. Ressaltamos que tanto esta etapa, quanto a etapa de Definição dos Planos de Execução, embora sejam elencadas por este trabalho como essenciais em um método para projeto de Aplicações Web de Integração de Dados, não constituem o foco de nossa pesquisa, o qual está sobre a questão da Especificação Conceitual.

3.3 Pré-requisito: Integração Semântica

Para que uma aplicação Web de integração de dados seja projetada utilizando o método proposto, é necessário que um pré-requisito seja atendido: a realização da integração semântica das fontes acessadas pela aplicação. Nesta seção, descrevemos como pode ser obtida uma Arquitetura de Integração Semântica.

Como visto no Capítulo 2, existem arquiteturas específicas para o a realização da Integração Semântica entre fontes de dados heterogêneas e distribuídas. Cada uma destas arquiteturas apresenta vantagens para determinados tipos de aplicação e são implementadas de formas distintas, utilizando diferentes níveis de ontologias e diferentes tipos de mapeamentos entre estas.

Visando a obtenção da arquitetura básica de Integração Semântica proposta no Capítulo 2 e a consequente implantação de um ambiente de Integração Semântica para a aplicação Web, esta etapa é composta por três passos, conforme ilustrado na Figura 3.1: (i)Geração da Ontologia de Aplicação, (ii) Geração das Ontologias Exportadas e (iii) Definição dos Mapeamentos OA-OEs.

Figura 3.1 – Diagrama de Atividades da etapa de Integração Semântica

3.3.1 Passo 1: Definição da Ontologia de Aplicação (OA)

A Ontologia de Aplicação (OA) é uma ontologia de domínio específico, que descreve os conceitos relativos a uma aplicação Web em particular, i.e., representa o esquema conceitual global de uma aplicação Web. Preferencialmente, deve basear-se em ontologias existentes utilizadas por comunidades específicas ou publicadas no Linked Data. Os princípios do Linked Data são aplicáveis a esta abordagem de definição de ontologia, uma vez que afirma que para facilitar o máximo possível para o cliente o processamento dos seus dados, devem ser reutilizados termos de vocabulários conhecidos, sempre que possível. É desejável que somente sejam definidos novos termos caso não seja possível encontrar os termos necessários em um vocabulário existente [Bizer et al. 2007]. Portanto, uma ontologia descrevendo os termos de uma aplicação de integração de dados deve ser definida principalmente com base em vocabulários padrão, e.g., DBpedia Ontology[10], Dublin Core Terms[11], FOAF[12], GeoNames[13], voiD[14] e assim por diante. No entanto, pode ser necessário usar termos fora do âmbito de vocabulários padrão. Nesses casos, o usuário pode definir seus próprios termos na Ontologia da Aplicação (OA).

Logo, a Ontologia da Aplicação pode ser obtida principalmente de duas formas:

1. **Uma ontologia é modelada especificamente para a aplicação** - neste caso, os termos da OA são definidos de acordo com os requisitos de conteúdo da aplicação, e deve ser basear-se primordialmente em um vocabulário existente, a ser estendido. A modelagem deverá ser realizada por um especialista do domínio. A OA será obtida a partir da definição das entidades, atributos, relacionamentos e regras de negócio identificadas no esquema conceitual da aplicação, da seguinte forma: as entidades do esquema conceitual são representadas, na ontologia, por classes; os atributos das entidades são representados por propriedades de dados; os relacionamentos entre entidades são representados por propriedades de objetos; e as regras de negócio da aplicação, por axiomas definidos na linguagem de ontologia utilizada.

[10] http://wiki.dbpedia.org/Ontology
[11] http://dublincore.org/documents/dcmi-terms/
[12] http://www.foaf-project.org/docs
[13] http://www.geonames.org/ontology/
[14] http://www.w3.org/TR/void/

Diversas ferramentas podem ser utilizadas para a modelagem de ontologias, tais quais Protégé, KAON, DOME[15].

2. **Uma ontologia pré-existente é utilizada** - caso já exista uma ontologia que descreve o domínio em que a aplicação está inserida aplicação, ou a área de conhecimento a que pertence, a OA pode ser representada como uma especialização desta ontologia. Dados os avanços das pesquisas sobre a Web Semântica, essa situação tende a ser a mais comum. Atualmente, diversos domínios, tais quais: jurídico, biomédico, educacional e organizacional dentre outros, são descritos semanticamente através de ontologias desenvolvidas por especialistas do domínio. Um exemplo de ontologia criada especificamente para uma aplicação, a partir de uma ontologia descrevendo uma determinada área de conhecimento, no domínio de Exploração e Produção de Petróleo, pode ser encontrado em [Lopes et al., 2008; Guizzardi et. al, 2009].

3.3.2 Passo 2: Geração das Ontologias Exportadas

Uma Ontologia Exportada descreve uma fonte de dados, utilizando o vocabulário comum da aplicação Web, i.e., a Ontologia da Aplicação. Há diversas abordagens na literatura sobre como gerar uma ontologia que descreve uma fonte de dados. Ressaltamos que quaisquer metodologias que obtenham este resultado podem ser utilizadas para a execução deste passo. Citamos como exemplo de uma abordagem eficiente para a geração de OEs em termos de uma OA a abordagem proposta em [Sacramento and Vidal 2010], a qual se baseia na Ontologia de Aplicação e nas Ontologias Locais, para a geração automática das Ontologias Exportadas.

[15] http://dome.sourceforge.net/

Segundo esta abordagem, para geração das Ontologias Exportadas, é necessária a geração prévia das Ontologias Locais, as quais são visões diretas das fontes de dados, expressas no vocabulário das mesmas. Para cada Ontologia Local, será gerada uma Ontologia Exportada equivalente. Para a geração das Ontologias Locais, deve ser utilizado o enfoque virtual, devido às vantagens citadas no Capítulo 2. Para isso, cada fonte local deve prover uma visão virtual RDF dos seus dados, a qual funciona como interface de acesso aos dados que residem nas fontes. A visão provida por cada fonte constitui sua respectiva Ontologia Local. No enfoque virtual, para possibilitar o acesso aos dados, uma consulta realizada em termos da ontologia é traduzida para uma consulta sobre o esquema da fonte, tendo como base um conjunto de mapeamentos entre ambas as estruturas.

A partir das correspondências identificadas em uma etapa de pós-matching, é induzido um conjunto de regras de mapeamento OL-OA utilizados na geração da Ontologia Exportada. O vocabulário da Ontologia Exportada consiste nas classes e propriedades que constituem o subconjunto da Ontologia de Aplicação que corresponde (matches) com a Ontologia Local, juntamente com as classes e propriedades da Ontologia de Aplicação, utilizadas para definir o vocabulário contextualizado da aplicação. Além disso, as restrições da Ontologia Exportada são a tradução das restrições da Ontologia Local, utilizando as regras de mapeamento OL-OA. Maiores detalhes sobre o processo de geração das OEs, pode ser encontrado em [Sacramento and Vidal 2010].

3.3.3 Passo 3: Definição de Mapeamentos OA-OE

Após a geração das Ontologias Exportadas, deve ser estabelecido um conjunto de regras de mapeamentos entre OA e OEs, identificando as correspondências entre conceitos da OA e das OEs, permitindo o desdobramento das consultas realizadas sobre a OA. A mesma abordagem citada no passo anterior prevê a geração automática destes mapeamentos, o que diminui a carga de serviço sobre o usuário. Entretanto, ressaltamos que também nesta fase quaisquer metodologias para o estabelecimento destes mapeamentos, inclusive manualmente, podem ser utilizadas.

De acordo com a abordagem de [Sacramento and Vidal 2010], é possível que seja identificada mais de uma correspondência para um mesmo conceito da Ontologia de Aplicação, o que implica a geração de mais de uma regra para este conceito. Desta forma, todas as regras de um mesmo conceito são reunidas em um corpo disjuntivo (uma lista de conjunções). As regras de mapeamento mediadas são diretamente obtidas a partir das regras OL-OA em forma disjuntiva, em conjunto com as regras OL-OE. As regras de mapeamento mediado podem ser utilizadas para o desdobramento de consultas submetidas sobre a Ontologia de Aplicação, em uma ou mais subconsultas sobre as Ontologias Exportadas [Vidal et al. 2009].

3.4 Considerações Finais

Neste capítulo definimos o método para o projeto e implementação de unidades de conteúdo dinâmico de aplicações Web de integração de dados. Explicamos as atividades pertinentes às etapas que compõem o método: Especificação Conceitual da Aplicação, Definição dos Planos de Execução e Publicação dos Dados, além de explicarmos detalhadamente o pré-requisito para a utilização deste método: a Integração Semântica. O emprego deste método apresenta diversas vantagens, entre as quais: aumento na qualidade da aplicação, devido à padronização; independência dos indivíduos, pois, conhecendo as etapas seguidas durante o desenvolvimento da aplicação, um desenvolvedor é capaz de realizar mais facilmente a manutenção em um sistema desenvolvido por terceiros; aumento da produtividade e gerenciamento eficiente do desenvolvimento. Mas principalmente, a possibilidade de especificar em alto nível de o conteúdo dinâmico de uma aplicação, de forma a possibilitar a automatização de sua implementação; além maior simplicidade na manutenção da aplicação Web, devido à sua implementação centralizada em um plano de execução gerado automaticamente.

4. Especificação Conceitual e Definição da Aplicação

Neste capítulo apresentamos detalhadamente a etapa de Especificação Conceitual da Aplicação (a qual constitui nossa contribuição) e a etapa de Definição dos Planos de Execução da aplicação. Na Seção 4.1 apresentamos uma aplicação exemplo, para melhor compreensão do método. Na Seção 4.2 detalhamos a etapa 1 do método: Especificação Conceitual, discutindo a definição de uma VCD e todos os passos necessários para defini-la, inclusive apresentando o algoritmo para geração automática da consulta. Por fim, na Seção 4.3 discutimos a etapa 2 do método, isto é, a Definição dos Planos de Execução, comentando cada um de seus passos.

4.1 Aplicação Exemplo

A etapa de Especificação Conceitual da Aplicação consiste em determinar os requisitos de conteúdo das unidades de conteúdo dinâmico da aplicação e, para cada unidade, especificar conceitualmente uma Visão de Conteúdo Dinâmico correspondente. A etapa de Definição dos Planos de Execução das VCDs consiste na geração do plano de execução da consulta que define a VCD e na disponibilização deste plano através de um mecanismo que permita o acesso ao usuário. Cada uma das atividades que compõem estas etapas será detalhada nas subseções deste Capítulo.

Para melhor compreensão destas atividades, consideremos como exemplo um estudo de caso (detalhado no Apêndice A), no domínio da saúde, que ilustra a concepção de um Mashup, cujas fontes são pertinentes ao Linked Data (as ontologias Diseasome, DailyMed, DrugBank, DBpedia e Sider), e, portanto, já possuem uma integração semântica definida e uma boa aceitação na comunidade. O mashup denominado Doenças e Drogas (D & D) integra dados sobre doenças e sobre as drogas utilizadas em seus respectivos tratamentos. Estes dados são obtidos de cinco fontes de dados que estão disponíveis publicamente na Web. D & D é composto por quatro páginas, que permitem a seguinte interação: o usuário pode procurar por uma doença, usando um critério definido (parâmetro), através da Home Page.

O resultado desta consulta é mostrado na Página 1 (NomesDoenças), que apresenta uma lista de nomes de doenças. Uma vez que o usuário seleciona um nome de doença, ele é direcionado para Página 2 (DrogasDoenças), que mostra uma lista de possíveis medicamentos para tratar a doença selecionada anteriormente. Finalmente, usuário seleciona um nome de medicamento (droga) e é redirecionado à Página 3 (DetalhesDrogas), que exibe detalhes sobre as drogas selecionadas. Desta forma, a partir de um parâmetro dado pelo usuário (nome de doença), o usuário consulta detalhes de uma droga relacionada à doença consultada, obtendo informações de diversas fontes. A Figura 4.1 mostra o modelo de hipertexto WebML, que especifica o *frontend* da aplicação.

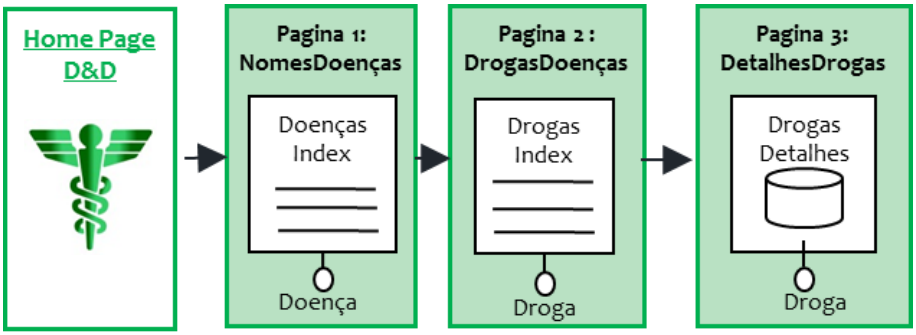


Figura 4.1 – Exemplo de aplicação Web de integração de dados: Mashup Doenças e Drogas

Como explicado no Capítulo III, o pré-requisito para a utilização do método é a realização de uma integração Semântica entre as fontes. Uma vez que nosso estudo de caso é realizado sobre Linked Data, consideramos que os *endpoints* dos *datasets* escolhidos já estão registrados na aplicação. A Figura 4.2 mostra a Ontologia da Aplicação projetada para o exemplo de aplicação utilizado neste Capítulo.

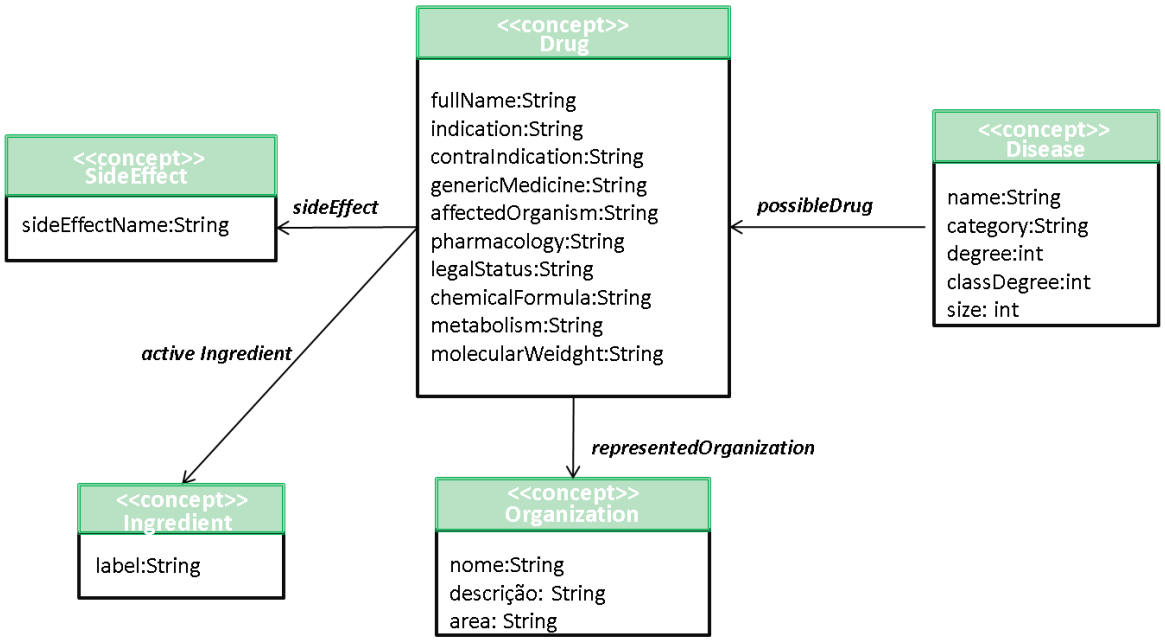


Figura 4.2 – Ontologia da Aplicação Mashup Doenças e Drogas

A ontologia apresentada representa o esquema conceitual da aplicação D&D. A Ontologia da Aplicação pode ser criada por meio de um editor de ontologias e deve basear-se primordialmente em um vocabulário existente. Neste exemplo, foi criada uma OA a partir de partes relevantes das seguintes ontologias, bem conhecidas do domínio de saúde: Diseasome, DailyMed, DrugBank, DBpedia e Sider. Estes são os *datasets* que serão consultados em nosso exemplo.

Um passo essencial para a escolha dos datasets que serão consultados por uma aplicação, especialmente no caso do Linked Data, é o registro. A atividade de registro consiste em escolher os conceitos apropriados, dentro de uma ontologia, que melhor descrevem um *dataset* [Bowers and Ludäscher, 2003]. Através da análise de metadados de *datasets* já registados, torna-se fácil selecionar os *datasets* relevantes para serem consultados.

Em nosso exemplo, uma vez que os *datasets* selecionados e a Ontologia da Aplicação estão relacionados a um vocabulário comum publicado em Linked Data, uma consulta sobre a OA pode ser reescrita diretamente sobre a ontologia exportada do *dataset* e apenas com a criação de mapeamentos simples R2R[16] entre a OA e as OEs (Diseasome, DailyMed, DrugBank, DBpedia e Sider), que constitui o passo de geração dos mapeamentos OA-OEs.

[16] www4.wiwiss.fu-berlin.de/bizer/r2r

Tal registro consiste em um mapeamento entre as estruturas de dados RDF locais (no caso de DBMS ou planilhas, este modelo é estabelecido pelo wrapper) para a estrutura de dados global, ou seja, a OA predefinida. O registro da fonte de dados pode ser gerenciado por um componente de registro, onde novas fontes de dados podem ser registradas em um repositório apropriado. Uma fonte de dados é registrada através do URI do *endpoint* SPARQL, fornecido pelo wrapper correspondente. O processo de registro do *dataset* baseia-se em uma tripla <OE, OS, M> onde: (i) OE é uma ontologia, que descreve o conjunto de dados, fornecido pelo *endpoint*, que será usado para fins de mapeamento do esquema. Esta ontologia é gerada muito semelhante ao esquema de dados local e, em seguida, é mapeada para OA; (ii) OS é uma ontologia fonte, já registrada no Linked Data, cujos conceitos corretamente podem descrever a OE; e (iii) m é um conjunto de mapeamentos OWL, criado a partir de regras de mapeamento EO e OS, definindo suas correspondências semânticas. Após o registo de uma nova fonte de dados, esta será considerada pelo mashup para consulta.

Após o registro, o monitor da fonte de dados tentará encontrar metadados na Web, que podem incluir: o mantenedor do *dataset*, informações de licença e tópicos de assuntos relacionados ao *dataset*. Um formato de metadados adotado é voiD[17], o " Vocabulary Of Interlinked Datasets ", um vocabulário que permite descrever formalmente *datasets* no Linked Data.

4.2 Etapa 1: Especificação Conceitual da Aplicação

A definição da etapa de Especificação Conceitual da Aplicação consiste em nossa principal contribuição dentro do método proposto, visto que apresentamos uma nova forma de especificar este conteúdo em alto nível e de forma a possibilitar a geração automática do plano de execução da consulta.

Os requisitos das unidades de conteúdo dinâmico de cada página de uma aplicação Web de integração de dados, devem ser especificados através de uma visão denominada Visão de Conteúdo Dinâmico (VCD). Para cada interação necessária entre o usuário e a aplicação, para a realização de uma tarefa que demande recuperação de conteúdo dinamicamente, há uma VCD correspondente. A etapa de Especificação Conceitual da Aplicação consiste em projetar todas as interações dos usuários, gerando um conjunto de VCDs que determina uma aplicação.

[17] http://www.w3.org/TR/void/

A dinamicidade das páginas de uma aplicação não é determinada somente pelo estado das fontes de dados envolvidas, mas também por um ou mais parâmetros cujos valores dependem da navegação realizada pelo usuário na aplicação; e.g., quando o usuário seleciona na página “NomesDoenças” da aplicação D&D um nome de doença, a aplicação o conduz para a página “DrogasDoença”, cujo conteúdo é o conjunto de drogas utilizadas no tratamento à doença cujo nome foi selecionado na página anterior, i.e., o conteúdo da página foi gerado de acordo com o parâmetro passado pelo link. Nesse sentido, a VCD é uma visão parametrizada, cujos parâmetros são definidos com base na troca de dados existente entre as páginas. Dessa forma, cada VCD é definida através de uma consulta parametrizada sobre a Ontologia da Aplicação.

A definição conceitual de uma VCD consiste em uma quádrupla $<\mathcal{S}, \mathcal{P}, Q, \mathcal{E}>$, a qual é obtida com base nas relações presentes na Infraestrutura Baseada em Ontologias para Especificação da Aplicação Web, apresentada no Capítulo II. Quanto aos termos presentes na quádrupla que define uma VCD:

(i) $\mathcal{S}$ é o esquema XML da visão, o qual define quais os atributos que constituem a VCD, determinando a estrutura do resultado obtido, conforme especificado no projeto da aplicação;

(ii) $\mathcal{P}$ é uma lista de parâmetros, os quais são definidos com base na troca de dados entre as páginas. Especifica os parâmetros utilizados na condição de seleção da consulta da VCD. Dessa forma, uma VCD é uma visão de seleção, que contém todos os objetos para os quais o valor de um determinado atributo x é igual ao atributo p, da lista de parâmetros $\mathcal{P}$;

(iii) Q é uma consulta parametrizada, escrita na linguagem SPARQL, que define a VCD sobre a OA. É gerada automaticamente, a partir de $\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$. Esta consulta permite a definição da VCD sobre o vocabulário comum da aplicação.

(iv) $\mathcal{E}$ é conjunto de elos que especificam relacionamentos navegacionais com outras VCDs, i.e., quais os parâmetros que devem ser utilizados para ligar uma VCD origem a uma VCD destino. Um elo é uma conexão orientada entre duas VCDs, relacionando os objetos de ambas;

Para possibilitar a especificação de uma VCD, i.e., possibilitar a definição precisa de seus atributos, há uma sequência de atividades (Figura 4.3), que constitui os passos da etapa de Especificação Conceitual da Aplicação. O objetivo desta sequência de passos é permitir que conjunto final de VCDs obtido, satisfaça aos requisitos de dados e funcionais que motivaram o desenvolvimento da aplicação.

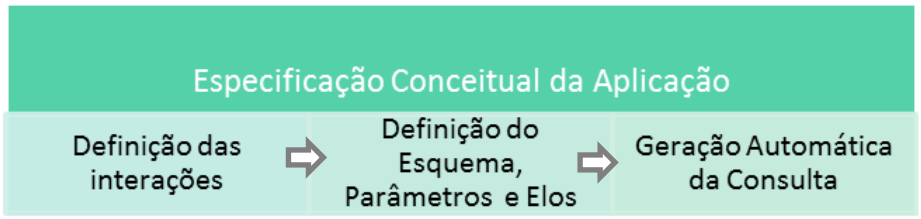


Figura 4.3 – Atividades da etapa de Especificação Conceitual

Após definida uma VCD, para representa-la fisicamente, um arquivo deve possuir as seguintes propriedades associadas a ele: nome da VCD (funciona como identificador único), URI do esquema da VCD, uma lista de parâmetros no formato `<tipo-parâmetro, nome-parâmetro>`, uma lista de elos no formato `<nome-elo, URI-VCD-destino>`. Como exemplo, representamos a estrutura básica de uma VCD através de um documento XML, como pode ser visto na Figura 4.4.

```
xs:schema
  xmlns:xs = "http://www.w3.org/2001/XMLSchema"
  xs:complexType
    name = "VCD"
    xs:sequence
      xs:simpleType
        name = "VCD_Schema"
        xs:restriction
          base = "xs:anyURI"
      xs:complexType
        name = "ParametersList"
        xs:sequence
          xs:simpleType
            nome = "parameter_name"
            xs:restriction
              base = "xs:parameter_type"
    xs:simpleType
      name = "Query"
      xs:restriction
        base = "xs:string"
    xs:complexType
      name = "NavigationalLinkList"
      xs:sequence
        xs:simpleType
          nome = "navigational_link_name"
          xs:restriction
            base = "xs:string"
```

Figura 4.4 – Exemplo de definição do XML Schema de uma VCD

4.2.1 Passo 1: Definição de Interações do Usuário

Em nossa proposta, o passo de Definição de Interações do Usuário constitui uma modelagem das interações do usuário com a aplicação durante o uso, identificando quais informações são solicitadas e enviadas pelo usuário a cada interação. Está relacionado às fases de Levantamento de Requisitos e Projeto Conceitual e Navegacional da aplicação Web, não abrangendo aspectos relativos ao Projeto de Interfaces. Adotamos parcialmente a abordagem proposta por [Guell et al, 2000], no que concerne ao método para permitir o levantamento de requisitos e capturar as interações necessárias entre o usuário e a aplicação Web. A proposta de [Guell et al, 2000] é baseada em três elementos:

(i) Cenários de Usuário [Carrol 1995], uma descrição textual ou narrativa de um episódio de uso. O cenário é descrito a partir do ponto de vista do usuário e pode incluir o fundo social, os recursos (espaço em disco, por exemplo, tempo), restrições e informações;

(ii) Casos de Uso [Jacobson et al., 1999], que são descrições de conjuntos de sequências executadas por um sistema, dentro de um cenário de uso, que levam a um resultado observável por um ator em particular;

(iii) Diagramas de Interação com Usuário [Vilain et al., 2000], uma ferramenta focada na interação, i.e, descreve apenas as trocas de informações entre os usuários e a aplicação, sem considerar aspectos relacionados com interface do usuário. Representam graficamente as interações textualmente descritas nos casos de uso.

Dentre estes, adotamos em nosso método o uso de Diagramas de Interação com o Usuário, por esta ferramenta descrever exatamente as interações necessárias para o projeto das VCDs. O desenvolvimento de uma aplicação Web tem como base as informações coletadas durante a atividade de Levantamento de Requisitos. As demais técnicas utilizadas para o levantamento de requisitos da aplicação ficam a critério do projetista, não sendo necessariamente definidas no método que propomos para o projeto de aplicações de integração de dados. Os requisitos são definidos durante os primeiros estágios de desenvolvimento de um sistema, como uma especificação da forma como deve ser implementado ou como um tipo de restrição no sistema [Sommerville 2000].

De acordo com [Schwabe and Rossi 1995], a atividade de Levantamento de Requisitos consiste nos seguintes passos: (i) identificação dos atores e tarefas a serem realizadas através da aplicação; (ii) especificação dos cenários que descrevem os passos da interação do sistema com o usuário, na execução de cada tarefa; (iii) especificação dos casos de uso, obtidos a partir do agrupamento de cenários que descrevem uma mesma tarefa.

A atividade de Levantamento de Requisitos é enriquecida pelo uso de Diagramas de Interação dos Usuários (UID), para representar as interações entre usuários e a aplicação durante a realização de uma tarefa descrita por um caso de uso. A proposta dos UID's é suprir as deficiências encontradas no emprego de casos de uso. Uma vez que são representados por descrições textuais, os casos de uso frequentemente causam ambigüidades, trazendo falta de precisão e concisão. Para cada caso de uso obtido é especificado um UID correspondente.

A Figura 4.5 mostra o UID "drogasPorDoença" especificado para a aplicação "D&D", onde há uma interação definida para cada página da aplicação.

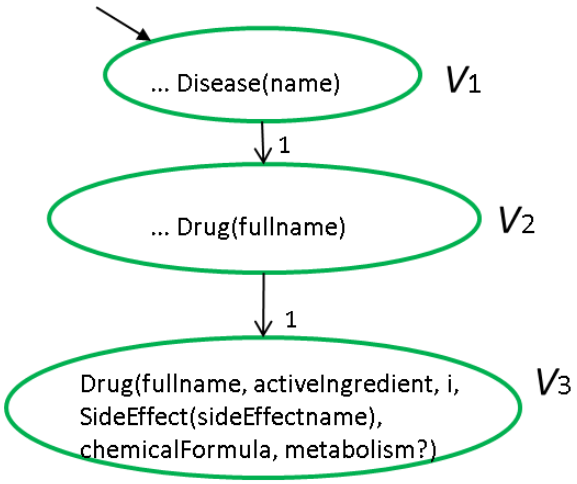


Figura 4.5 – UID "drogasPorDoença", pertinente à aplicação D&D

Observa-se pelo UID da aplicação que a interação V1 é relativa a uma página que exibe uma lista de nomes de Doenças. Ao selecionar uma das doenças exibidas, o usuário é direcionado para a página da interação V2, na qual é exibida uma lista de Drogas relacionadas à doença selecionada, onde para cada droga, exibe-se o nome completo da droga. Ao selecionar uma das drogas da lista, o usuário é direcionado para a página da interação V3, onde se exibem os detalhes da droga selecionada: nome completo, princípio ativo, nome dos efeitos colaterais, a fórmula química e o metabolismo (optativamente).

Em nossa proposta, para cada interação identificada em um UID, deverá ser projetada uma VCD que realize esta interação, permitindo que ao fim da interação o usuário receba como resultado os requisitos de conteúdo identificados para tal interação. Desta forma, tomaremos como exemplo a interação V2 do UID apresentado, para os próximos passos apresentados no método.

4.2.2 Passo 2: Definição de Esquema ($\mathcal{S}$), Parâmetros ($\mathcal{P}$) e Elos ($\mathcal{E}$)

Para cada interação definida em um UID deve ser projetada uma VCD, a qual especifica os requisitos de conteúdo no contexto dessa interação. Durante o projeto de uma VCD, devem ser especificados os valores para os campos da quadrupla <$\mathcal{S}$, $\mathcal{P}$, Q, $\mathcal{E}$> que define uma VCD sobre a Ontologia da Aplicação. Nesta subseção descrevemos como se dá a obtenção destes campos, exceto da consulta Q, a qual será gerada automaticamente em um passo posterior, a partir dos demais campos definidos.

Na definição do esquema $\mathcal{S}$ da VCD devem ser levados em consideração quais os dados a serem exibidos como resultado e qual a estrutura esperada para apresentação desse resultado. Estes atributos são obtidos a partir dos requisitos de conteúdo definidos no contexto da interação do UID em questão, i.e., cada requisito de conteúdo corresponde a um atributo. Dessa forma, o esquema XML padrão de uma VCD deve conter basicamente os seguintes elementos: um elemento base, o qual representa uma classe da Ontologia de Aplicação a partir da qual são selecionados os atributos que compõem a VCD; o conjunto de atributos (obrigatórios e opcionais) no formato `<atributo, tipo>` e uma condição de ordenação, no formato `<atributo, operador, valor>`, que define a ordem em que os resultados devem ser exibidos, caso haja alguma. Dessa forma, o esquema $\mathcal{S}$ de uma VCD, representado em XML, deve seguir o padrão exibido a seguir, na Figura 4.6.

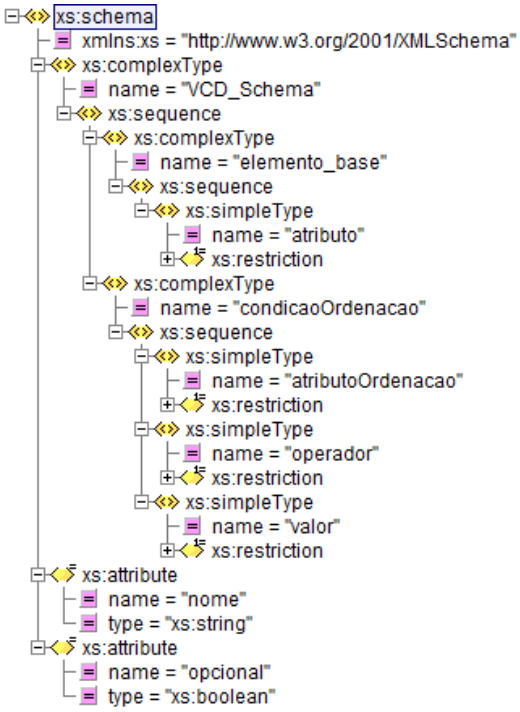


Figura 4.6 – Schema XML padrão ($\mathcal{S}$) de uma VCD

Utilizando como exemplo nosso estudo de caso, a definição do esquema da interação V2 do UID drogasPorDoença é exibida a seguir, na Figura 4.7. O esquema representa fielmente os requisitos de conteúdo definidos em V2, i.e., o nome completo da droga selecionada, que é um atributo obrigatório e não há condição de ordenação. O elemento Drugs é o elemento sobre o qual se obtém informações, embora a partir deste elemento seja possível atingir outros, como o SideEffect na interação V3. Entretanto, deseja-se obter somente os nomes das drogas relacionadas a uma determinada doença, portanto, neste esquema o elemento base definido é Drugs.

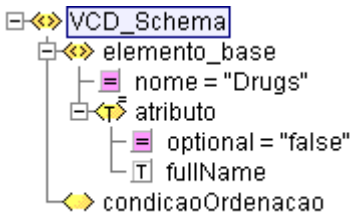


Figura 4.7 – Definição de $\mathcal{S}$ para a interação V2

Para a definição da lista $\mathcal{P}$ de parâmetros da VCD, consideramos as associações pertencentes a uma interação do UID em questão. Cada associação (que representa uma troca de dados entre páginas) possui a identificação do parâmetro utilizado como link entre as páginas, de forma que para cada parâmetro de uma associação pertencente a uma interação, é definido um parâmetro para a VCD deste UID.

Em nosso exemplo, o parâmetro de transição da interação V1 para a interação V2 é o nome de uma doença, visto que a interação V2 exibe somente as drogas que tratam a doença selecionada pelo usuário. Logo, a lista de parâmetros de V2 pode ser definida como:

$$P = \{s: string \mid s \text{ é } o\ nome\ de\ uma\ Doença\}$$

Para a definição da lista $\mathcal{E}$ de elos entre as VCDs, cada associação entre as interações de um UID define um elo entre as VCD deste UID. Cada associação representa uma troca de dados entre as páginas. A utilização de elos permite a passagem de parâmetros entre VCDs. Um elo deve ser definido sob a forma:

$$\text{nome\_elo} : \{\text{Destino: [VCD\_destino]; Parâmetros: [param\_x]}\}$$

Em nosso exemplo, o parâmetro de transição da interação V2 para a interação V3 é o atributo `fullName` de uma droga, visto que a interação V3 exibe os detalhes somente da droga selecionada pelo usuário. Logo, a lista de elos de V2 pode ser definida como:

$$E = \{detalhesDroga\}, \qquad \text{onde:}$$

$$detalhesDroga : \{\text{Destino: [V3]; Parâmetros: [fullName]}\}$$

4.2.3 Passo 3: Geração Automática da Consulta Q

A consulta Q que permite a definição da VCD sobre a ontologia da aplicação é gerada automaticamente, utilizando a linguagem SPARQL, tomando como base os campos definidos no passo anterior, i.e., os atributos, as condições, os parâmetros e as correspondências definidas em $\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$. Além disso, internamente, o algoritmo de geração da consulta utiliza mapeamentos entre os parâmetros e atributos da VCD e as propriedades da OA.

Tais mapeamentos são um conjunto $\mathcal{M}$ de assertivas de correspondência e definem um mapeamento conceitual entre o esquema da VCD e a Ontologia da Aplicação, explicitando quais as propriedades da Ontologia da Aplicação que correspondem aos atributos definidos no esquema $\mathcal{S}$. Esta definição de mapeamentos somente se faz necessária caso os nomes dos parâmetros sejam distintos dos nomes utilizados nas propriedades da OA. Um arquivo de assertivas de correspondência é basicamente composto por um conjunto de elementos "correspondência", o qual contém o atributo do esquema da VCD e a propriedade correspondente na OA. O conjunto de assertivas de correspondência deve conter as correspondências entre atributos e propriedades no formato `Correspondência <atributoVCD, propriedadeOA>` como mostrado no XML Schema a seguir (Figura 4.8). Em nosso caso, utilizamos os mesmos nomes de atributos, pelo que não é necessário especificar este mapeamento.

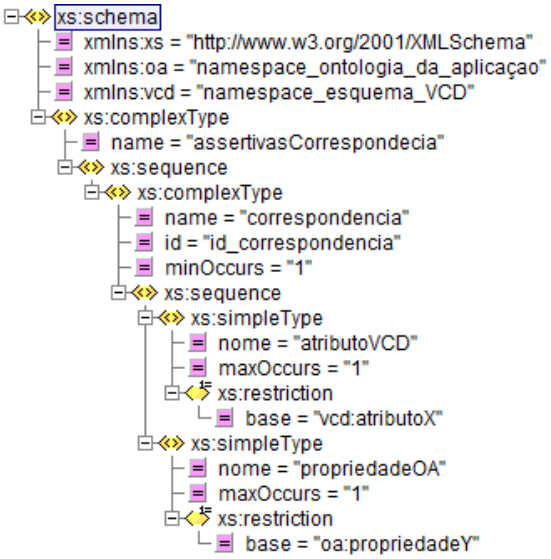


Figura 4.8 – Schema XML padrão do arquivo de assertivas de correspondência

A consulta gerada com base nos elementos apresentados utiliza a linguagem SPARQL que, segundo a Recomendação SPARQL do W3C, é a linguagem de consulta da Web Semântica, e pode ser utilizada para expressar consultas em diversas fontes de dados, nas quais os dados são armazenados nativamente em RDF, ou visualizadas como RDF através de um middleware. Uma consulta SPARQL inclui, em ordem:

- Cláusula PREFIX: Declarações de prefixo, para a abreviação de URIs;
- Cláusula FROM: Definição do conjunto de dados (dataset), declarando quais os grafos RDF (ontologias) a serem consultados;

- Cláusula SELECT: Uma cláusula de resultado, identificando quais as informações (ou as variáveis) que devem ser apresentadas no resultado da consulta;
- Cláusula WHERE: O padrão da consulta, especificando o que consultar no dataset;
    - No escopo da cláusula WHERE, pode ocorrer a cláusula FILTER, a qual expressa uma restrição sobre o conjunto de dados, através da filtragem no valor de uma variável; e/ou a cláusula OPTIONAL, a qual permite que parte do padrão da consulta seja opcional.
- Cláusula ORDER BY, dentre outras: Modificadores de consulta, que permitem rearranjar os resultados da consulta.

Para a geração de Q_3 consideramos: os atributos definidos em $\mathcal{S}$, como os atributos da projeção de Q (cláusula SELECT); a consulta será realizada sobre a Ontologia da Aplicação (cláusula FROM) e o mapeamento $\mathcal{M}$ e a lista $\mathcal{P}$ especificam quais as condições e os parâmetros utilizados na condição de seleção da consulta da VCD (cláusulas WHERE e FILTER). A geração da consulta é realizada a partir da execução do algoritmo de geração de consulta SPARQL exibido a seguir (Algoritmo 4.1). Caso não seja definido um mapeamento entre VCD e OA, por padrão utilizam-se correspondências por nome dos elementos. Há uma restrição quanto à ordem em que as correspondências são definidas em $\mathcal{M}$: a ordem dos elementos "correspondência" em $\mathcal{M}$ é obtida seguindo o mesmo padrão de uma busca em profundidade no grafo da Ontologia da Aplicação. As primeiras correspondências inseridas em $\mathcal{M}$ são aquelas que pertencem ao elemento base; em seguida, todas as correspondências pertencentes a cada classe que se relaciona à classe base; e assim sucessivamente, até que se retroceda à classe base.

Uma limitação para a definição destes mapeamentos é que não são permitidos ciclos entre as correspondências identificadas. Tanto a restrição de ordem das correspondências quanto a limitação relativa aos ciclos, são importantes para a utilização do algoritmo de geração da consulta Q_3

A Consulta 1 exibe a consulta correspondente para a definição da VCD V2, gerada segundo descrito acima.

```
Q = :  ({?drName },
          ((?x rdf:type oa:Disease) AND (?x oa:name ?name) AND
          (?y rdf:type oa:Drug) AND (?y oa:fullName ?drName)AND
          (?x oa:relatedTo ?y)
     FILTER regex (?name,“s") )))
```

Consulta 1 – $\mathcal{Q}$: define V2 sobre a OA da aplicação Doenças&Drogas

O algoritmo **GeraConsulta** recebe como entrada o esquema XML ($\mathcal{S}$), a lista de parâmetros ($\mathcal{P}$) e os mapeamentos ($\mathcal{M}$) de uma VCD e a Ontologia da Aplicação e retorna a consulta $\mathcal{Q}$ que permite a definição da VCD sobre a Ontologia da Aplicação. Este algoritmo é exibido e explicado no Apêndice B. A Figura 4.9 exibe o resultado da VCD *drugsIndex* projetada para a interação V2 do UID *drogasPorDoença*, da aplicação D&D, conforme todos os passos mostrados para a etapa de Especificação Conceitual da Aplicação.

VCD V2 *drugsIndex*	
$\mathcal{S}$: **Attribute List**	*fullName: String*
$\mathcal{P}$: **Parameter List**	*name: String*
$\mathcal{Q}$: **Query**	*PREFIX md: <http://www.lia.ufc.br/medical/resource/medical>* *?drName ,* *?x rdf:type oa:Disease AND ?x oa:name ?name AND* *?y rdf:type oa:Drug AND ?y oa:fullName ?drName AND* *?x oa:relatedTo ?y* *FILTER regex ?name,“s"*
$\mathcal{E}$: **Navigational Links**	*DrugsMoreInfo: {Destiny: {drugsDetails};* *Parameters:{fullName}}*

Figura 4.9 – Representação da VCD drugsIndex

4.3 Etapa 2: Definição dos Planos de Execução

Como discutido anteriormente, uma VCD é definida conceitualmente como uma consulta SPARQL sobre a Ontologia da Aplicação. Para a execução desta consulta, é necessário que haja um plano de execução correspondente. Um Plano de Execução de uma consulta consiste em um conjunto ordenado de passos utilizados para acessar a informação contida em uma fonte de dados.

De acordo com a Infraestrutura de Integração Semântica apresentada, quando uma consulta Q é executada sobre a Ontologia de Aplicação, é necessário que esta seja decomposta (reescrita) em um conjunto de subconsultas elementares sobre as Ontologias Exportadas, de forma que cada subconsulta extraia dados de uma única Ontologia Exportada. Portanto, é necessário que o plano de execução de Q defina, eficientemente, a ordem e a combinação para as subconsultas de Q.

A etapa de Definição dos Planos de Execução consiste em gerar um plano de execução para uma dada consulta associada à uma VCD. Esta seção descreve a sequência de operações desta etapa. Um plano de execução contendo a sequência otimizada de operações para integrar dados deve ser gerado a partir de uma consulta SPARQL parametrizada (Q). Essa geração ocorre em tempo de projeto e permite otimizações em consultas que não poderiam ser feitas em tempo de execução, devido a restrições de tempo.

Para definir esta etapa do nosso método (Figura 4.10) nos baseamos nos passos propostos por [Pinheiro et al., 2009], o qual trata do problema de responder eficientemente a consultas, i.e., como computar a resposta a consultas submetidas sobre uma ontologia, que representa uma visão integrada, em uma arquitetura de integração semântica. Em nosso trabalho, a OA representa uma visão integrada em uma arquitetura de integração semântica e as consultas submetidas sobre esta ontologia são as consultas que definem as VCDs da aplicação Web.

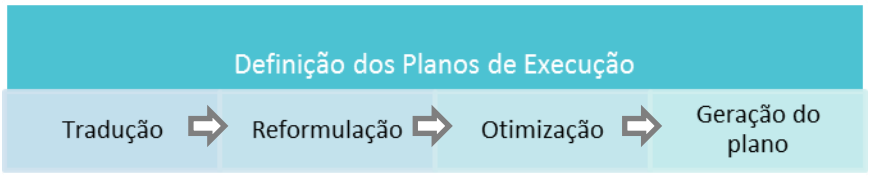


Figura 4.10 - Atividades da etapa de Definição dos Planos de Execução

1. Tradução. A consulta SPARQL parametrizada (Q) definida sobre a OA é convertida em um plano de consulta em álgebra SPARQL.

2. Reformulação. O plano de consulta em álgebra SPARQL é reformulado em um conjunto de subconsultas sobre as OEs. O plano resultante é chamado de plano de consulta federado. Os mapeamentos definidos entre as ontologias, em conjunto com serviços de *reasoning* de *Description Logics* (DL), são utilizados para gerar este plano que é uma combinação (uniões e junções) de subconsultas sobre as ontologias exportadas relevantes, onde cada subconsulta é definida utilizando a álgebra SPARQL. O plano de consulta federado especifica como os resultados obtidos das subconsultas são combinados para uma obtenção imediata do resultado. Toda a informação utilizada para responder à consulta inicial é descoberta neste passo.

3. Otimização. Uma série de operações é realizada sobre o plano de consulta federado para otimizá-lo visando boa performance na execução da consulta e redução dos custos de transmissão de dados remotos. O passo de otimização é essencial, visto que os dados das fontes são acessíveis via Web, o que pode fazer com que a quantidade de dados retornados seja desconhecida, afetando a performance da rede.

4. Geração do Plano de Execução. O plano de consulta federado otimizado em álgebra SPARQL é compilado para a geração do plano de execução com as operações necessárias para que a partir dos parâmetros da consulta, sejam realizadas as sub-consultas sobre as fontes de dados relevantes e a construção do resultado final da consulta.

Para exemplificar, utilizamos a consulta parametrizada referente à interação V3 da aplicação D&D, por se tratar de uma consulta mais complexa em que podem ser visualizadas as propriedades desta etapa. A consulta Q_3está associada à VCD projetada para a interação V3 e recupera detalhes sobre uma droga. A consulta Q_3 é traduzida para um plano de consulta em álgebra SPARQL que é então reformulado em um plano de consulta federado Q_3" (Figura 4.11), o qual define subconsultas sobre os *datasets* Dailymed, DrugBank, Sider e DBpedia. Após isto, o plano de consulta federado Q_3" é otimizado para gerar o plano de consulta Q_3"', o qual é finalmente compilado para um plano de execução E_3. O plano de execução E_3 define todas as operações necessárias para obter os resultados da consulta Q_1 gerada automaticamente na etapa 2.

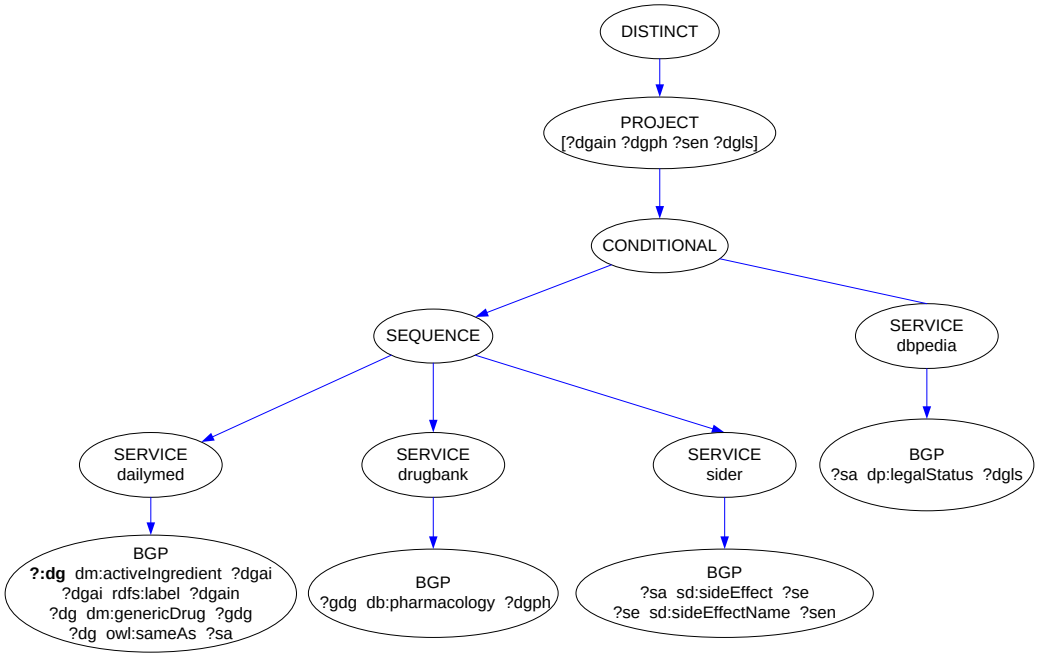


Figura 4.11 - Plano de Consulta Federado Q_3"

Para execução do plano final obtido, um ambiente de execução apropriado deve ser utilizado. Uma opção de mecanismo de execução de consultas disponível é o framework QEF *(Query Evaluation Framework)* [Porto et al., 2007], o qual poderá ser estendido, em um trabalho futuro, para trabalhar com a álgebra e os operadores SPARQL, uma vez que originalmente lida eficientemente com processamento de consultas paralelas, entretanto, utilizando XML.

4.4 Considerações Finais

Neste capítulo consideramos as duas primeiras etapas do método proposto para projeto e implementação de aplicações Web de integração de dados. Focamos e detalhamos a questão do uso de VCDs como forma de padronizar o projeto do conteúdo dinâmico deste tipo de aplicação. Através do uso de VCDs, dentro do método proposto, é possível especificar tal conteúdo em alto nível, visto que não há necessidade de conhecer os esquemas das diversas fontes a serem acessadas, nem de se utilizar diversas APIs e linguagens de programação. Além disso, possibilita-se, pela definição da consulta parametrizada, a geração automática dos planos de execução das consultas, o que leva a um planejamento prévio da aplicação, o que facilita sua manutenção e possibilita uma execução otimizada da aplicação.

5. Publicação dos Dados

Este capítulo apresenta, na Seção 5.1, conceitos sobre anotação e publicação de dados na etapa 3 do método proposto. Na seção 5.2 falamos sobre Serviços Web de Acesso a Dados, mecanismo sugerido para a publicação de dados na Web. Em seguida, na Seção 5.3, falamos sobre anotação semântica de serviços Web, discutindo as vantagens e desvantagens das diversas abordagens utilizadas atualmente. Na Seção 5.4 expomos uma forma de como estes conceitos podem ser utilizados em nossa proposta e discutimos a questão da encontrabilidade dos serviços Web que dão acesso às VCDs, tornando-as públicas. Por fim, na Seção 5.5 abordamos os cenários de acesso a uma aplicação Web desenvolvida utilizando nosso método.

5.1 Anotação e Publicação de VCDs

A terceira etapa do método proposto neste trabalho é referente à publicação dos dados e é composta pelos passos de anotação semântica e publicação de VCDs (Figura 5.1). Propomos anotar semanticamente cada VCD de uma aplicação Web através de sua definição <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$>, com base na Ontologia da Aplicação, e disponibilizar esta VCD através de um Serviço Web de Acesso a Dados, permitindo que clientes externos reutilizem as VCDs de uma aplicação.

O uso de anotações semânticas, para a descrição de serviços Web, tem se mostrado uma solução eficiente para este problema, uma vez que suplanta a necessidade de acordos prévios com clientes em potencial e alivia ou elimina a discrepância terminológica [Verma and Sheth 2007].

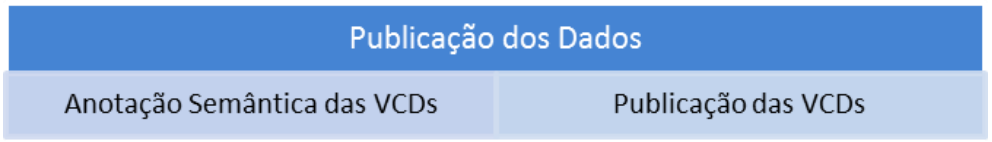


Figura 5.1 – Passos da etapa de Publicação de Dados

Além disso, quando modelos formais – como ontologias – são utilizados na anotação de serviços, as anotações se tornam processáveis por máquina, o que constitui um quesito essencial para a automação de tarefas na Web Semântica. Descreveremos a seguir os conceitos de Serviços Web de Acesso a Dados, discutiremos as formas atuais de anotar um serviço semanticamente e, por fim, apresentamos uma proposta de anotação e publicação dos serviços das VCDs que viabilize uma busca semântica eficiente por tais artefatos.

5.2 Serviços Web de Acesso a Dados

Segundo o W3C Web Services Activity[18], serviços Web constituem uma classe particular de serviços, a qual provê meios padronizados para interoperação entre diferentes aplicações de software, que podem ser executadas em uma variedade de plataformas e frameworks. Serviços Web são caracterizados por sua interoperabilidade e extensibilidade, assim como por suas descrições processáveis (*machine-processable*), obtidas graças ao uso de XML. Estes serviços podem ser combinados, de forma fracamente acoplada, possibilitando a execução de operações complexas.

De forma abrangente, serviços Web podem ser compreendidos como aplicações, com as quais outras aplicações podem estabelecer comunicação, através de uma rede, geralmente utilizando XML [Kaye 2003]. De maneira mais formal, um serviço Web é definido como uma aplicação autocontida, identificada por um Uniform Resource Identifier (URI), onde as interfaces e ligações são definidas, descritas e localizadas por artefatos que utilizam a linguagem XML [AUSTIN et al, 2002]. Tais artefatos são: a própria linguagem XML para especificação de conteúdo, Web Services Description Language (WSDL) para descrição de serviço, e Universal Description Discovery and Integration (UDDI) para implementação de serviços de diretório.

[18] www.w3.org/2002/ws/

É possível enumerar diversas vantagens trazidas pelo uso de serviços Web: interoperabilidade a baixo custo, principalmente por não assumir nenhum detalhe específico sobre a implementação das aplicações clientes; redução da complexidade por meio do encapsulamento de interfaces e componentes; e ainda, prolongamento da utilização de aplicações legadas. Devido a essas vantagens, os serviços Web têm sido amplamente utilizados como mecanismo interoperável de extração e acesso a dados armazenados em fontes distintas.

Quando utilizados dessa forma, os serviços são denominados Serviços Web de Acesso a Dados (SWAD). Tais serviços permitem a publicação de visões, a partir de dados armazenados em uma fonte. Esta estratégia oferece uma forma transparente e flexível para publicar dados, além de encapsular detalhes de acesso e recuperação entre aplicações e dados. Propomos o uso desta estratégia para a publicação das VCDs definidas para uma determinada aplicação. Desta forma, cada VCD de uma aplicação Web desenvolvida segundo nosso método, pode ser disponibilizada publicamente na Web, de forma que os dados recuperados por estas possam ser reutilizados por diversas aplicações.

Diversos trabalhos definem formas de acesso a fontes de dados, através do uso de serviços Web. Na área geográfica, onde o compartilhamento de informações é importante, devido ao alto custo de produção e a complexidade dos dados espaciais, os serviços Web de Acesso a Dados são bastante adotados. O Open Geospatial Consortium[19] (OGC) define as recomendações de padronização que garantem a interoperabilidade entre Sistemas de Informações Geográficas (SIGs) na Web através de serviços Web de Acesso a Dados. Duas importantes iniciativas do OGC são: a Geography Markup Language[20] (GML) e a especificação Web Feature Service[21] (WFS). A GML é uma linguagem baseada em XML, considerada o formato padrão para representação de dados geográficos na Web. A especificação WFS, por sua vez, tem o propósito de descrever as operações (métodos) WFS de manipulação de dados codificados em GML.

[19] www.opengeospatial.org
[20] www.opengeospatial.org/standards/gml
[21] www.opengeospatial.org/standards/wfs

O trabalho de [Costa 2009] propõe a especificação Web XML Service (WXS), a qual provê uma interface padrão para acesso a dados relacionais no formato XML na Web. A especificação WXS é baseada na especificação WFS e define os métodos: (i) getCapabilities, permite aos usuários solicitar e recuperar a descrição dos metadados ou arquivo de capacidades, que são os documentos que descrevem as habilidades da implementação específica do servidor; (ii) getViewType, permite ao usuário recuperar a descrição da estrutura de qualquer visão XML (um esquema XML); e (iii) query, permite ao usuário recuperar instâncias de uma visão XML usando a linguagem XQuery. Com base na especificação definida pelo WFS para serviços Web de acesso a dados, e na interface padrão proposta por [Costa 2009], para acesso a dados XML, percebe-se a possibilidade de generalização da definição de serviços Web de acesso a dados, para prover outros tipos específicos de visões das fontes.

5.3 Anotação Semântica de Serviços Web

Os serviços Web trazem um novo nível de funcionalidades para a Web, no sentido de contribuir para a criação de um ambiente aberto de aplicações distribuídas. Entretanto, duas dificuldades podem ser detectadas, quanto ao uso destes serviços: (i) embora as tecnologias atuais, sobre as quais os serviços Web são tradicionalmente construídos, formem uma base sólida, a escalabilidade dos serviços é limitada, caso não haja o devido grau de automação [Kopecky et al. 2007]; e (ii) para compreender e utilizar um determinado serviço disponível na Web, um cliente deve compreender a semântica de cada operação do serviço, i.e., o cliente deve ser capaz de decifrar inequivocamente o propósito de cada operação, bem como o conteúdo de cada parâmetro [Verma and Sheth 2007].

Com relação à necessidade de automação, requer-se mais do que a descrição estrutural e sintática, provida pelas descrições XML dos serviços Web, para possibilitar a automação do uso destes serviços. Segundo [Berners-Lee et al. 2001], a automação pode ser ampliada através do uso das tecnologias semânticas, propostas pela iniciativa da Web Semântica. Quanto à necessidade de compreensão da semântica do serviço, segundo [Verma and Sheth 2007], existem pelo menos três abordagens, através das quais os provedores de serviços buscam fornecer esta compreensão aos clientes: (i) estabelecimento de um pré-acordo, entre provedores de serviços e clientes, sobre todos os termos e parâmetros utilizados no serviço; (ii) inserção de comentários sobre todos os aspectos de um serviço e (iii) anotação dos elementos de um serviço, utilizando termos provenientes de modelos de domínio, o que inclui padrões industriais, vocabulários, taxonomias e ontologias.

As especificações mais proeminentes, que visam acrescentar semântica aos serviços Web, são OWL-S[22], WSMO[23], WSDL-S[24] e mais recentemente, a recomendação do W3C SAWSDL[25], a qual evoluiu da especificação WSDL-S. Segundo [Menegazzo et al. 2009] uma análise destas linguagens demonstra que WSMO é a linguagem mais completa quanto à semântica, entretanto, é considerada a mais complexa devido á necessidade de um ambiente de execução próprio (WSMX). Por sua vez, SAWSDL é a linguagem mais limitada quanto à semântica, porém, sua vantagem está em sua praticidade, pois seu documento de descrição é semelhante a um WSDL, eliminando a necessidade de adaptações no processo de descrição. Por fim, OWL-S, quanto à utilização dos conceitos semânticos e a praticidade, representa o papel de meio-termo dentre as linguagens analisadas. Através dela, pode-se descrever grande parte dos recursos de um serviço Web, mas não de forma tão completa quanto WSMO. Quanto à sua praticidade, OWL-S não é tão complexa quanto WSMO, pois não necessita de um ambiente de execução próprio, porém apresenta maior complexidade que SAWSDL, pois necessita de uma máquina de execução específica para processar seus documentos `.owl`.

A finalidade primordial da descrição semântica de serviços Web é permitir a descoberta destes serviços, por agentes de software, especialmente no contexto da Web Semântica. Segundo [Fensel et al. 2002], descoberta de serviços Web é um critério de avaliação para as linguagens de descrição semântica de serviços. Este critério apresenta como cada linguagem realiza a descoberta dos serviços, demonstrando quais elementos das linguagens são aplicados para realizar buscas aos serviços e como a Web semântica está embutida nessas buscas. Entretanto, de acordo com a análise realizada por [Menegazzo et al. 2009] cada linguagem semântica citada necessita utilizar ferramentas específicas, que funcionam conforme a composição de seus elementos, para permitir a descoberta de serviços Web descritos em determinada linguagem.

Em [Lutz 2005] é proposta uma metodologia para a descrição de serviços Web, que permite a descoberta destes serviços sem a dependência de ferramentas específicas ou a necessidade de ambientes de execução próprios. A metodologia proposta provê a descrição dos serviços baseada em ontologias e o uso de mecanismos de *matchmaking* entre as descrições dos serviços, para aumentar a precisão durante a descoberta de serviços para acesso e processamento de dados.

[22] www.w3.org/Submission/OWL-S
[23] www.wsmo.org
[24] www.w3.org/Submission/WSDL-S
[25] www.w3.org/2002/ws/sawsdl

As descrições semânticas propostas na metodologia de [Lutz 2005] são diferenciadas de acordo com o tipo de serviço a ser descrito, da seguinte forma. (i) Abordagem Baseada em Ontologias para Descoberta de Serviços de Acesso a Dados - baseada na descoberta de correspondências semânticas entre conceitos representando recursos de dados (descritos através de Description Logics) e a consulta realizada pelo agente; (ii) Abordagem Baseada em Ontologia para Descoberta de Serviços Processamento de Dados – ontologias são utilizadas para a descrição de operações de processamento de dados, bem como na criação de descrições dos requisitos de usuário (ambas as descrições em DL). Um mecanismo baseado em subtipagem de funções é utilizado para determinar quando a descrição de um serviço corresponde à descrição solicitada pelo usuário. Nas duas abordagens, as descrições de serviços são baseadas em um vocabulário compartilhado (ontologia), que contém os termos básicos do domínio.

5.4 Anotação e Publicação de VCDs como Serviços Web de Acesso a Dados

Estamos de acordo com [Lutz 2005] quanto ao fato de que uma metodologia que provê descrições baseadas em ontologias e mecanismos de *matchmaking* baseados nestas descrições leva a uma melhor precisão na descoberta e reutilização de serviços para acesso e processamento de dados. Desta forma propomos, com base na metodologia de [Lutz 2005], que cada VCD seja descrita utilizando os conceitos específicos da aplicação para a qual foram projetadas, utilizando as regras e conceitos pertencentes à Ontologia da Aplicação. Desta forma, quando uma VCD apropriada é descoberta pelo usuário, a consulta associada a esta VCD pode ser utilizada para gerar uma solicitação de recuperação dos dados.

Consideramos que os serviços de acesso a dados que dão acesso às VCDs estão registrados em catálogos de serviços Web. Para manter comparáveis as diversas descrições de VCDs contidas em um ou mais catálogos, é crucial que tais descrições estejam baseadas em um vocabulário compartilhado entre as diversas fontes, neste caso, a Ontologia da Aplicação. Portanto, a descrição do serviço em si é dada pela tripla $<\mathcal{S}, \mathcal{P}, \mathcal{Q}>$ em termos da OA, possibilitando encontra-la através da busca em um catálogo ou via agentes inteligentes, dados os atributos desejados pelo usuário, os parâmetros e/ou a consulta desejada, em termos da Ontologia da Aplicação, o que possibilita ao usuário utilizar termos mais amplamente difundidos e padronizados. A Figura 5.2 mostra o esquema de como essa busca semântica por VCDs pode ser realizada.

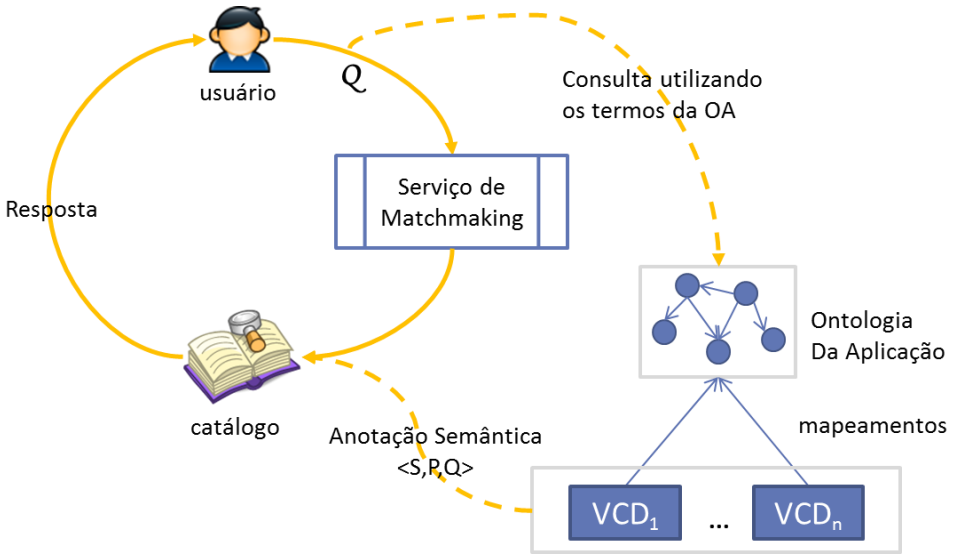


Figura 5.2 – Esquema de busca semântica por VCDs

Após um usuário selecionar uma dentre as diversas VCDs descobertas, o valor do parâmetro utilizado em sua busca é passado para a consulta parametrizada da respectiva VCD e a consulta é realizada sobre a OA e desdobrada sobre as OEs necessárias através dos mapeamentos entre estas. Como um passo final, a consulta é executada seu resultado retornado ao usuário.

Adotamos, assim, uma forma semanticamente eficiente para a publicação e descoberta de VCDs como serviços Web de Acesso a Dados, sem a dependência de ferramentas específicas ou a necessidade de ambientes de execução próprios. Entretanto, é importante ressaltar que não desenvolvemos o algoritmo capaz de realizar o *matchmaking* entre os conceitos da OA buscados pelo usuário e a descrição de uma VCD, ficando esta atividade para um trabalho futuro.

5.5 Cenários de Acesso à Aplicação Web

O acesso aos dados de uma aplicação Web desenvolvida com base no método proposto pode ocorrer de três maneiras distintas, as quais serão apresentadas a seguir.

Cenário 1: Acesso Interno. O cenário mais comum de acesso aos dados de uma aplicação Web, em nosso contexto, é caracterizado pela presença de um usuário acessando a aplicação através de uma interface Web. Neste cenário, no momento em que o usuário interage com a aplicação solicitando conteúdo de uma unidade de conteúdo dinâmico, a aplicação identifica a VCD correspondente aos dados solicitados e seleciona o Plano de Execução correspondente à VCD. Cada Plano de Execução contém diversas subconsultas, cada uma definida sobre uma única fonte. Tais subconsultas são orquestradas sobre as diversas fontes, segundo a sequência definida pelo PE. Cada fonte possui um tipo de wrapper o qual fornece acesso às fontes de dados. Podem ser utilizados como *wrappers* mecanismos desenvolvidos especificamente para o acesso à fonte, ou Serviços Web de Acesso a Dados. Os wrappers recebem as subconsultas, enviam-nas para execução nas fontes e retornam seus resultados para a aplicação, que integra os resultados recebidos e os retornam para serem apresentados na interface da aplicação Web.

Cenário 2: Acesso Externo, por agentes. Para permitir que os dados disponíveis em uma aplicação estejam acessíveis para os agentes inteligentes da Web Semântica, um serviço que permite o acesso à aplicação é registrado em um catálogo, utilizando como descrição termos de uma Ontologia de Domínio, mapeados na Ontologia da Aplicação. Dessa forma, quando agentes da Web Semântica buscam por serviços que coincidem com a descrição semântica de uma VCD, é possível informar ao agente se os dados solicitados por este podem ser obtidos a partir da aplicação à qual pertence a VCD em questão e então, iniciar o processo de materialização destes dados.

Neste cenário, quando um agente na Web Semântica deseja descobrir um serviço de dados que forneça o conteúdo de que necessita, o mesmo realiza uma consulta em termos de uma Ontologia de Domínio (OD), a qual constitui um vocabulário compartilhado, que descreve os termos básicos do domínio sobre o qual o agente atua. Neste caso, deseja-se descobrir se algum dos serviços, que foram registrados no catálogo utilizando a mesma OD sobre a qual a consulta do agente foi definida, é capaz de responder à consulta do agente.

Seguindo a abordagem de [Lutz 2005] para a descoberta semântica de serviços de acesso a dados, a partir da consulta do agente sobre a OD, é gerada automaticamente uma consulta de conceitos em Description Logics e, através de *reasoning*, é determinado se existem conceitos na Ontologia da Aplicação que correspondem aos conceitos da consulta do agente. Se não existirem conceitos correspondentes na Ontologia da Aplicação, significa que nenhuma VCD da aplicação é capaz de materializar os dados solicitados pelo agente.

Entretanto, se os conceitos estiverem presentes na OA, é necessário checar se alguma VCD é capaz de materializar todos os conceitos requeridos pela consulta. Então, a consulta realizada pelo agente é enviada para um serviço que recebe como entrada uma consulta e, através de um algoritmo de *pattern matching* [Sedgewick and Wayne 2010], determina se a consulta é implementada por alguma das VCDs da aplicação. Em caso afirmativo, o serviço envia o identificador da VCD correspondente à consulta do agente. A partir daí, o processo de acesso aos dados ocorre semelhantemente ao descrito no cenário 1.

Cenário 3: Acesso Externo, por Mashups. Um dos objetivos deste trabalho é facilitar a criação de Mashups. Neste cenário, possibilita-se que aplicações Mashup acessem transparentemente os dados contidos nas fontes integradas, sem a necessidade do uso de APIs específicas ou de técnicas de extração de dados. Entretanto, o acesso aos dados das fontes está restrito ao conteúdo recuperado pelas consultas que definem as VCDs da aplicação.

Este cenário assemelha-se ao cenário 2, exceto pelo fato de que não é necessário que o Mashup realize a descoberta semântica do serviço, para acessar a aplicação, uma vez que no desenvolvimento de um Mashup já se tem o conhecimento de quais as aplicações que serão utilizadas para compor a nova aplicação. Neste caso, a descrição semântica do serviço será utilizada para que o desenvolvedor da aplicação compreenda a semântica das operações do serviço. Deste modo, o típico esforço de programação que a criação de um Mashup demanda, pode ser aliviado, uma vez que para acesso a esta aplicação, é necessário somente o envio da consulta Q para o serviço correspondente à aplicação. A partir daí, o processo de acesso aos dados ocorre semelhantemente ao descrito no cenário 2.

5.6 Considerações Finais

As informações disponibilizadas por aplicações Web podem ser compartilhadas através de serviços Web, promovendo o objetivo da Web Semântica, i.e., permitir a execução de atividades baseadas em informação. Neste capítulo tratamos sobre Serviços Web de Acesso a Dados, discutindo como estes podem ser aplicados ao nosso trabalho. Oferecemos uma motivação para o uso de anotações semânticas em serviços Web, inserindo estes serviços no contexto da Web Semântica. Por fim, discutimos sobre como VCDs podem ser publicadas e descobertas através de descrições baseadas em ontologias, permitindo assim a reutilização de VCDs projetadas para uma determinada aplicação Web de integração de dados.

6. Framework Conceitual e Protótipo de Ferramenta

Neste capítulo apresentamos um framework conceitual, o qual determina quais os componentes de software que devem ser utilizados para possibilitar a execução de todas as etapas do método proposto.. Ao longo deste capítulo descrevemos o conjunto de módulos e artefatos que compõem o Framework. Na Seção 6.1 apresentamos um esquema do framework, assim como descrições referentes a cada módulo, bem como descrevemos o Repositório utilizado para armazenar os artefatos produzidos e consumidos pelos módulos do framework. Na Seção 6.2 apresentamos o protótipo de uma ferramenta para instanciar o framework conceitual e por fim, na Seção 6.3 descrevemos a implementação do protótipo no que diz respeito à Especificação Conceitual.

6.1 Visão Geral do Framework Conceitual

Segundo [Johnson 1997], um framework pode ser definido como um design, em grande escala, que descreve como um sistema de software pode ser decomposto em um conjunto de partes, que interagem entre si. O propósito de um framework é definir um “esqueleto” de uma aplicação, que poderá ser customizada por um desenvolvedor, o qual decidirá os módulos e ferramentas que desempenharão os papéis de cada uma das partes, que compõem o framework.

Definimos neste trabalho um ambiente ou framework (Figura 6.1) composto por um conjunto de módulos, que oferece suporte a cada etapa do método proposto, bem como os artefatos gerados e consumidos em cada uma destas etapas. Para isso, o framework oferece suporte a cada passo das etapas de Integração Semântica, Especificação Conceitual, Definição dos Planos e Publicação de Dados. Cada módulo definido é responsável pela execução automática ou semiautomática das atividades que compõem cada etapa do método.. Os artefatos utilizados como entradas e saídas destes módulos são armazenados em um Repositório de Artefatos.

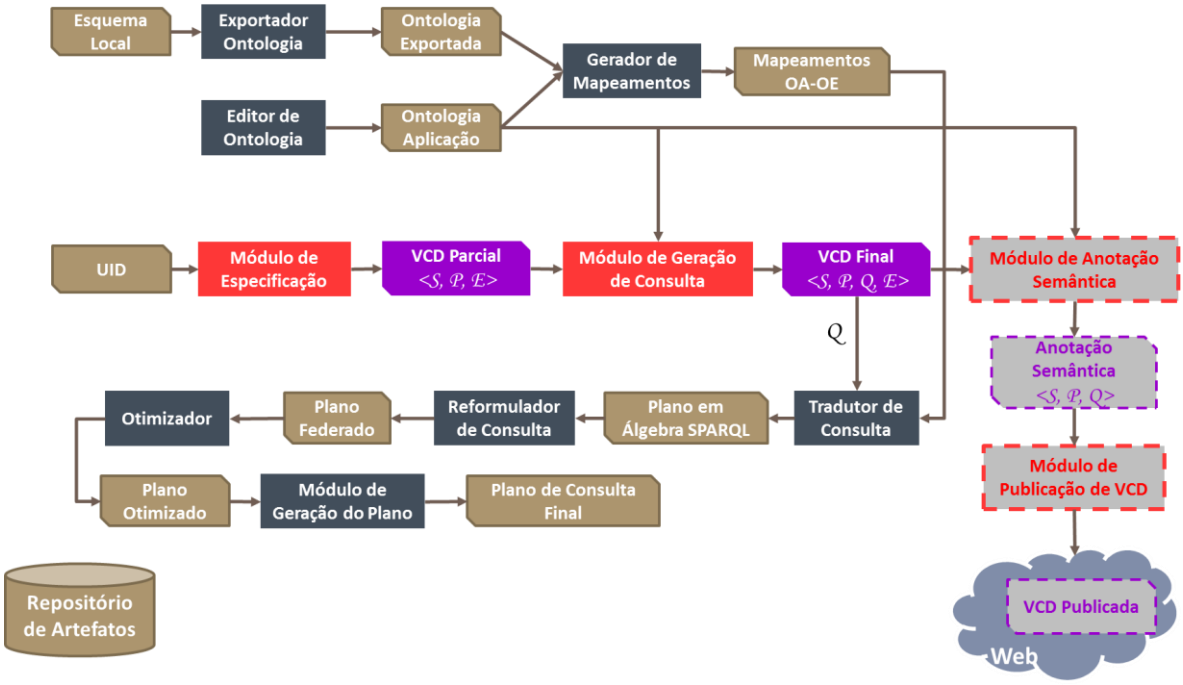


Figura 6.1 – Módulos e artefatos do Framework Conceitual

Visto que nossos esforços se concentram na etapa de Especificação Conceitual da Aplicação, sendo a definição da mesma contribuição nossa, os módulos relativos a esta etapa foram definidos e desenvolvidos por nós: Módulo de Especificação e Módulo de Geração de Consulta, assim os artefatos consumidos e gerados por tais módulos (especificação parcial e final da VCD e consulta SPARQL). Tais módulos encontram-se em destaque na Figura 6.1. Os módulos relativos à etapa de Publicação de Dados não foram desenvolvidos neste trabalho, entretanto, no Capítulo V discutimos uma forma eficiente de implementação da anotação semântica e publicação de VCDs. Quanto aos demais módulos, citaremos outros trabalhos que podem eficientemente aplicados para sua implementação. Nas subseções a seguir detalharemos as interações, módulos e artefatos presentes no framework.

6.1.1 Módulos de Integração Semântica

A primeira camada do framework visa dar suporte à Etapa de Integração Semântica, pré-requisito para aplicação de nosso método, a qual é composta pelas seguintes atividades: Definição da Ontologia da Aplicação, Geração das Ontologias Exportadas e definição dos mapeamentos OA-OEs. Dessa forma, esta camada é composta por um três módulos, cada um responsável pela execução de uma destas atividades: Editor de Ontologia, Exportador de Ontologia e Gerador de Mapeamentos.

1. *Exportador de Ontologia:* é o módulo responsável pela obtenção das Ontologias Exportadas correspondentes aos esquemas das fontes locais a serem integradas. Recebe como entrada o esquema de uma fonte de dados e retorna como saída uma ontologia RDF correspondente. Uma forma de implementar este módulo seria através da abordagem proposta em [Sacramento and Vidal 2010], a qual se baseia na Ontologia de Aplicação e nas Ontologias Locais (mapeamento direto com a fonte), para a geração automática das Ontologias Exportadas.

2. *Editor de Ontologia:* é o módulo responsável pela obtenção da Ontologia da Aplicação. Uma ontologia preexistente pode ser editada, ou uma nova ontologia pode ser criada, de acordo com os requisitos da aplicação. Exige a interação de um especialista de domínio e gera como saída uma ontologia RDF correspondente ao Esquema Conceitual da Aplicação Web.

3. *Gerador de Mapeamentos:* Recebe como entrada uma OA e um conjunto de OEs e define o mapeamento semântico entre os termos destas ontologias, gerando como saída um arquivo contendo este mapeamento. Uma possível forma de implementar este módulo seria através do método e do formalismo para representação de mapeamentos proposto por [Sacramento and Vidal, 2010]. A definição dos mapeamentos é feita de maneira semiautomática, isto é, com a intervenção de um especialista de domínio para validar os mapeamentos gerados (etapa de pós-matching).

6.1.2 Módulos de Especificação Conceitual da Aplicação

O segundo conjunto de módulos do framework visa dar suporte à Etapa de Especificação Conceitual da Aplicação, proposta pelo método apresentado neste trabalho, a qual é composta pelas seguintes atividades: Definição de Interações do Usuário, Definição de $\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$ e Geração Automática de Q. Dessa forma, esta camada é composta por um dois módulos, cada um responsável pela execução de uma destas atividades: Módulo de Especificação e Módulo de Geração de Consulta. Adiante neste capítulo apresentamos a ferramenta que implementa tais módulos.

1. *Módulo de Especificação:* é o módulo responsável pela definição dos termos de uma VCD, a partir da modelagem das interações do usuário, em termos de UIDs. Exige a interação do projetista da aplicação para modelar os UIDs e, a partir das interações presentes em cada UID, definir o Esquema XML ($\mathcal{S}$), os Parâmetros ($\mathcal{P}$) e os Elos ($\mathcal{E}$). Dessa forma, estes três artefatos constituem a saída do Módulo de Especificação. Para a implementação deste módulo deve ser desenvolvida uma ferramenta que permita a modelagem dos UIDs e a determinação de $\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$ de forma gráfica e simples.

2. *Módulo de Geração de Consulta* - responsável pela geração automática da consulta SPARQL que define a VCD sobre a OA. Recebe como entrada a especificação conceitual parcial de uma VCD ($\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$) e a OA, e gera automaticamente a consulta SPARQL Q sobre a OA, a qual constitui a saída deste módulo. A geração da consulta é realizada de acordo com o algoritmo GeraConsulta mostrado no Capítulo 4.

6.1.3 Módulos de Definição dos Planos de Execução

O terceiro conjunto de módulos do framework visa dar suporte à Etapa de Definição dos Planos de Execução, a qual consiste na geração automática dos planos de execução das consultas (Q) que definem as VCDs da aplicação. Dessa forma, esta camada é composta por quatro módulos: Tradutor de Consulta, Reformulador de Consulta, Otimizador e Módulo de Geração do Plano. Uma forma possível para implementação destes módulos é através da abordagem proposta por [Pinheiro et al, 2010].

1 *Tradutor de Consulta* – módulo responsável por traduzir a consulta parametrizada Q em um plano de consulta em álgebra SPARQL.

2 *Reformulador* – módulo responsável por reformular o plano de consulta em álgebra, em um plano de consulta federado, que define um conjunto de subconsultas sobre as OEs.

3 *Otimizador* – módulo encarregado de realizar as operações necessárias para otimizar o plano de consulta federado, visando melhoria na performance da consulta.

4 *Módulo de Geração do Plano* – realiza a compilação do plano federado, para a geração automática do plano de execução da consulta Q de uma VCD, contendo as operações necessárias para que, a partir dos parâmetros da consulta, sejam realizadas as subconsultas sobre as fontes de dados relevantes e a construção do resultado final da consulta.

6.1.4 Módulos de Publicação de Dados

O quarto conjunto de módulos do framework visa dar suporte à Etapa de Publicação de Dados, que consiste em anotar semanticamente cada VCD de uma aplicação Web através de sua definição <S, P, Q, E>, com base na Ontologia da Aplicação, e disponibilizar esta VCD através de um Serviço Web de Acesso a Dados, permitindo que clientes externos reutilizem as VCDs de uma aplicação. Esta camada é composta por dois módulos: Módulo de Anotação Semântica e Módulo de Publicação de VCDs. Uma forma possível para implementação destes módulos é através de uma adaptação da abordagem proposta por [Lutz 2005].

1 *Módulo de Anotação Semântica* – com base na definição final de uma VCD e na Ontologia na Aplicação, este módulo associa uma descrição a uma VCD, o que permite que esta VCD seja encontrada através de buscas por atributos, parâmetros e/ou consulta desejada, em termos da Ontologia da Aplicação.

2 *Módulo de Publicação de VCDs* – responsável por, dada a descrição semântica de uma VCD, publicá-la em um catálogo de serviços Web, em forma de um Serviço Web de Acesso a Dados, de forma que um serviço de *matchmaking*

possa verificar a similaridade entre os conceitos consultados por um usuário e os conceitos publicados como anotação de uma VCD.

6.1.5 Repositório de Artefatos

Além dos módulos citados, o framework proposto conta com um Repositório de Artefatos, utilizado para centralizar o armazenamento dos artefatos produzidos e consumidos pelos módulos do framework, tais quais esquemas, ontologias, especificações de VCDs, UIDs, mapeamentos, VCDs e serviços.

Este repositório deverá ser implementado através de um sistema de controle de versão, tal qual Subversion [26], o qual é um sistema centralizado para compartilhamento de informações. O Repositório de Artefatos é essencialmente um repositório central de dados, que armazena informações sob a forma de um sistema de arquivos em árvore, uma típica hierarquia de arquivos e diretórios. Qualquer número de clientes pode se conectar ao repositório, sendo capazes de ler ou escrever nos arquivos armazenados. Ao escrever dados, um cliente torna a informação disponível para os outros; ao ler dados, o cliente recebe informações inseridas ou modificadas por outros clientes. No framework proposto, os clientes do Repositório são os módulos que interagem para a execução do método e os arquivos armazenados constituem os artefatos consumidos e produzidos durante este processo.

O Repositório de Artefatos deverá ainda utilizar propriedades para armazenar informações adicionais (metadados) sobre arquivos, diretórios, e revisões. Portanto, no Repositório de Artefatos, os diretórios e arquivos serão descritos por metadados versionados, o que permitirá a busca mais eficiente dos artefatos armazenados.

O uso de um sistema de controle de versão para a implementação do Repositório de Artefatos traz a vantagem de garantir a coerência e a consistência dos artefatos utilizados pelos módulos componentes do framework, além de facilitar a organização e o acesso aos mesmos.

[26] subversion.apache.org

6.2 Protótipo de Ferramenta para Suporte ao Método

Realizamos ainda neste trabalho o projeto e a prototipação de uma ferramenta que permita o gerenciamento de VCDs para uma aplicação Web, de acordo com o modelo proposto pelo framework conceitual apresentado. É importante ressaltar que realizamos o projeto da ferramenta por completo, mas implementamos apenas a parte relativa aos módulos de Especificação e Geração da Consulta, juntamente com o gerenciamento (criação, edição e remoção) dos artefatos gerados e consumidos por estes módulos.

O objetivo desta ferramenta é agrupar um conjunto de tecnologias, capazes de executar as funções requeridas para os módulos do framework. Esta ferramenta deverá ser disponibilizada como um plugin do editor de ontologias Protégé[27]. Dessa forma, poderão ser reutilizadas as funções de manipulação de ontologias disponibilizadas pelo Protégé, para executar parte das atividades da etapa de Integração Semântica. Além disso, facilitaremos a difusão da ferramenta proposta, dentro da comunidade científica.

Durante o projeto da ferramenta, realizamos uma análise de requisitos, seguida da representação das funcionalidades do sistema através de casos de uso (Apêndice C). Finalmente, realizamos o levantamento de quais tecnologias podem ser utilizadas para a implementação de cada funcionalidade identificada (Apêndice D) e projetamos a interface gráfica da ferramenta. Uma vez que nos restringimos a exibir o protótipo da ferramenta, apresentamos nas seções seguintes os resultados obtidos na fase de projeto, bem como a implementação realizada até então, a saber: das funcionalidades relativas à etapa de Especificação Conceitual da Aplicação.

6.2.1 Projeto da Interface

Nesta seção exibimos o projeto da interface gráfica da ferramenta, projetada de forma conduzir ao seu uso simples e transparente, permitindo a aplicação prática do método proposto para o projeto de aplicações Web de integração de Dados.

[27] protege.stanford.edu

A Figura 6.2 apresenta a tela inicial da ferramenta, onde a área 1 representa o Painel de Navegação, através do qual o usuário tem acesso aos diretórios e arquivos do Repositório de Artefatos; a área 2 representa o Painel Central, onde estão localizadas as abas “Integração Semântica”, “Especificação da VCD” e “Implementação da VCD”. Nesta área ocorrem as interações relativas a cada uma destas atividades; por fim, a área 3 representa o Painel de Propriedades, o qual permite a visualização e edição das propriedades dos elementos selecionados no Painel Central.

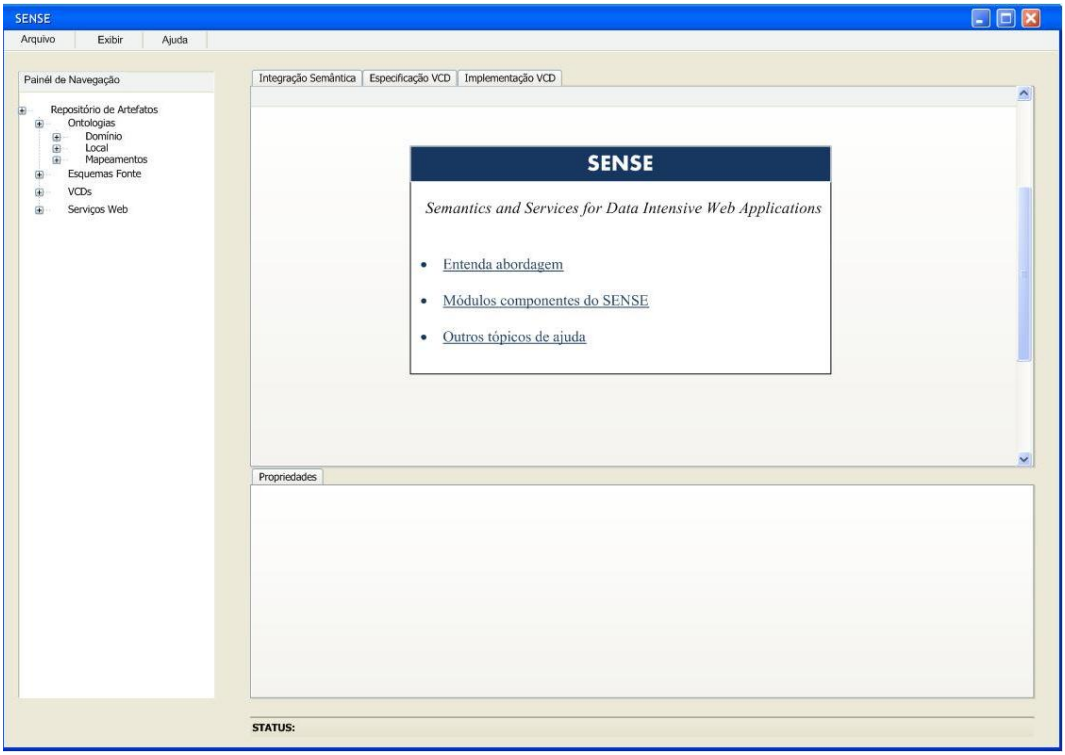


Figura 6.2 – Tela Inicial da ferramenta

A Figura 6.3 apresenta a tela da ferramenta durante a etapa de Integração Semântica. Observa-se a aba “Integração Semântica” ativa no Painel Central e as funcionalidades desta etapa estão disponíveis em uma barra de tarefas localizada abaixo da aba selecionada. Observam-se as opções: “Carregar Esquema”, “Extrair Ontologia”, “Gerar OE” e “Definir Mapeamentos”. No Painel Central encontra-se esquematizada a arquitetura de dois níveis da integração semântica. Observa-se a seleção da Ontologia da Aplicação e a exibição das propriedades relativas a este elemento, no painel de propriedades.

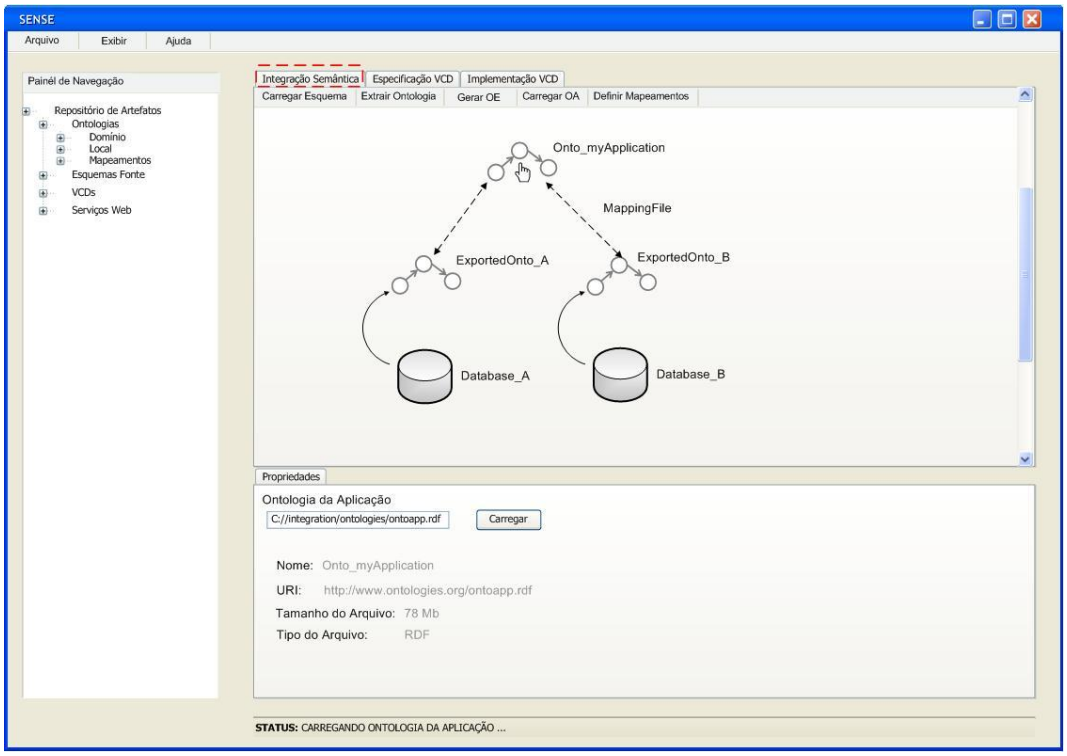


Figura 6.3 – Tela da Etapa de Integração Semântica

A Figura 6.4 apresenta a tela da ferramenta durante a etapa de Especificação Conceitual da VCD. Observa-se a aba “Especificação VCD” ativa no Painel Central e as funcionalidades desta etapa estão disponíveis em uma barra de tarefas localizada abaixo da aba selecionada. Observam-se as opções: “Novo UID”, “Criar VCD”, “Editar VCD” e “Excluir VCD”. No Painel Central encontra-se a representação gráfica das VCDs definidas para aplicação, bem como os relacionamentos entre estas. Observa-se a seleção de uma das VCDs e a exibição das propriedades relativas a este elemento, no painel de propriedades, as quais são baseadas na definição conceitual de uma VCD: <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$>.

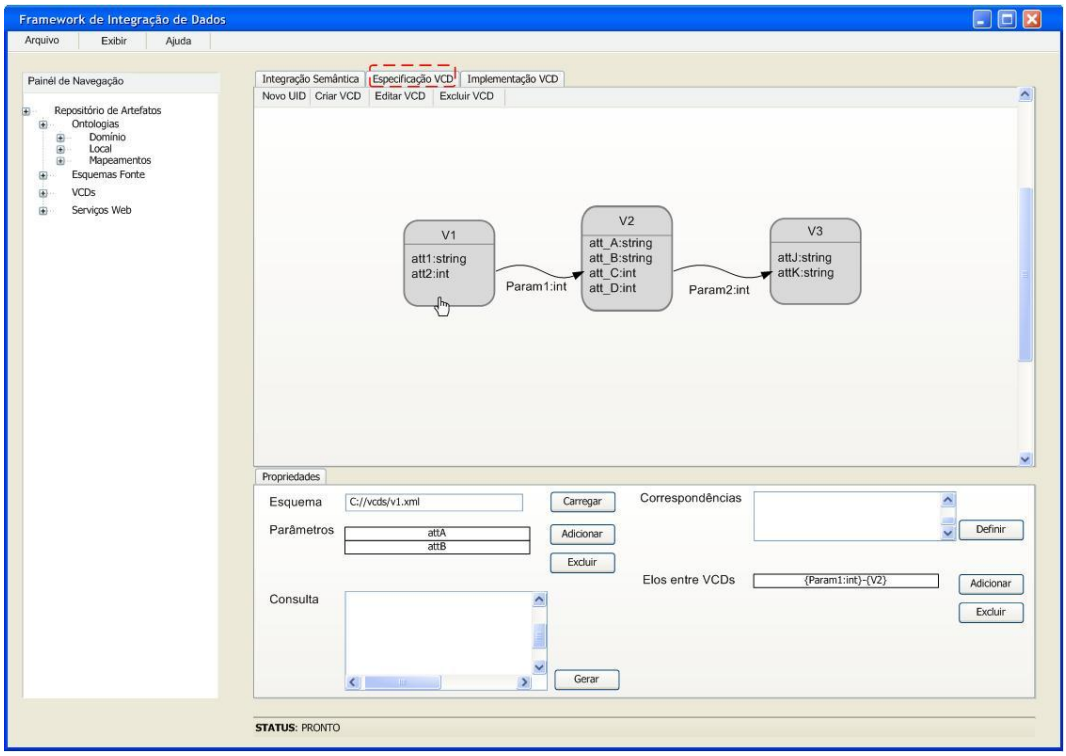


Figura 6.4 – Tela da Etapa de Especificação Conceitual da VCD

Os botões disponíveis na interface na Aba de Implementação da VCD dão início às etapas da geração dos planos de consulta das VCDs e disponibilizam opções relativas à anotação semântica e publicação, como: escolha dos metadados e do catálogo a ser publicado.

6.3 Implementação do Protótipo

Uma vez que a principal contribuição deste trabalho é a definição do método e a formalização da especificação conceitual de uma VCD, esta foi a etapa selecionada para ser implementada, no protótipo da ferramenta. A implementação das demais etapas propostas pelo método apresentado constitui um trabalho futuro a este.

Desta forma, encontra-se atualmente funcional a atividade de Criar uma VCD, definida em termos de <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$ >, onde o esquema $\mathcal{S}$ é definido como um arquivo XML; os parâmetros $\mathcal{P}$ como uma lista de parâmetros; a consulta $\mathcal{Q}$ como um documento SPARQL, gerado automaticamente a partir de $\mathcal{S}$, $\mathcal{P}$ e $\mathcal{E}$; e por fim, os elos $\mathcal{E}$ são definidos como uma lista de relacionamentos entre as VCDs. O modelo dos arquivos XML que representam estes termos e a própria especificação final da VCD, bem como o algoritmo de geração automática de $\mathcal{Q}$ são apresentados e discutidos no Capítulo 4.

A implementação destas funcionalidades foi realizada segundo discutido anteriormente, de forma que a Figura 6.5 apresenta o Diagrama de Classes da Implementação realizada.

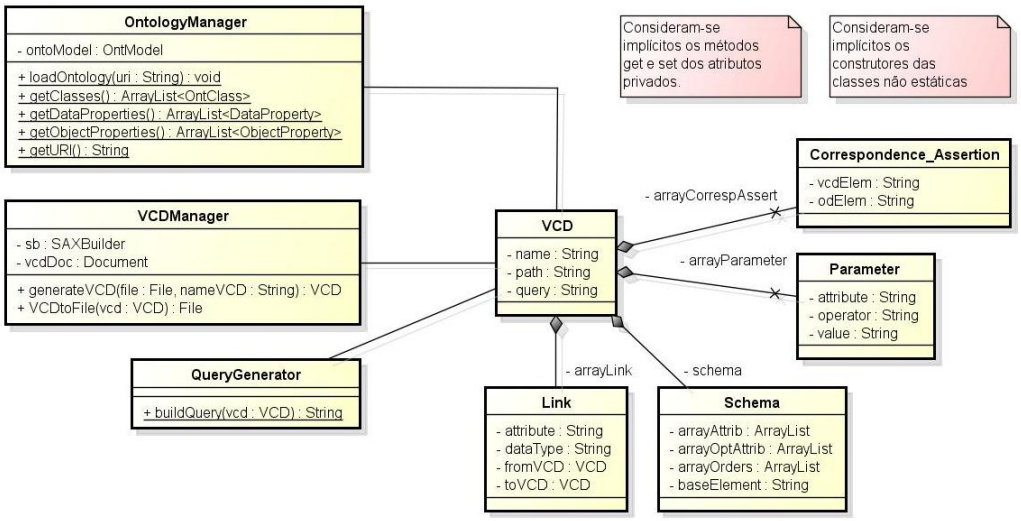


Figura 6.5 – Diagrama de Classes da implementação da atividade de Criação da VCD

O Diagrama apresentado é composto pelas seguintes classes:

- **OntologyManager**: é responsável pela manipulação da Ontologia da Aplicação. Para isso, esta classe possui cinco métodos estáticos que permitem a extração de informações sobre as classes da ontologia, suas propriedades de dados e propriedades de objetos.

- **VCD**: contém os atributos necessários para a especificação conceitual de uma VCD. Um objeto VCD possui um atributo query e um conjunto de composições de objetos das classes **Correspondence_Assertion**, **Parameter**, **Schema** e **Link**, o que permite a especificação de uma VCD de acordo com a quádrupla $<\mathcal{S},\mathcal{P}, Q, \mathcal{E}>$.

    - *O esquema* $\mathcal{S}$: corresponde ao atributo `schema` na classe VCD, o qual é um objeto **Schema** especificado pelos atributos: (i) `arrayAttrib` e `arrayOptAttrib` são listas de atributos a serem projetados na consulta Q, obrigatórios e opcionais, respectivamente; (ii) `arrayOrders`, especifica quais atributos devem ser utilizados para ordenar o resultado da consulta Q sobre a OA; e (iii) `baseElement` especifica qual a classe base da Ontologia de Aplicação, especificada pelo usuário.

- *A lista de parâmetros* $\mathcal{P}$: corresponde ao atributo `arrayParameter` na classe **VCD**, que é um uma lista de objetos **Parameter**. Cada objeto **Parameter** é especificado por uma tripla <`atributo`, `operador`, `valor`>.

- *Consulta* Q *em SPARQL*: corresponde ao atributo `query` na classe **VCD**, o qual é obtido através da execução do método `buildQuery`, da classe **QueryGenerator**.

- *Conjunto de elos* $\mathcal{E}$: corresponde ao atributo `arrayLink` na classe **VCD**, o qual é uma lista de objetos **Link**. Um objeto da classe **Link** é uma 4-tupla <`atributo`, `tipo`, `vcdOrigem`, `vcdDestino`>, onde a conexão entre `vcdOrigem` e `vcdDestino` é definida por `atributo`.

- Além da especificação da quádrupla <$\mathcal{S}$, $\mathcal{P}$, Q , $\mathcal{E}$>, a classe **VCD** contém os atributos `name` (nome do arquivo XML de especificação da VCD) e `path` (caminho da localização deste arquivo). Por definição padrão, o atributo `arrayCorrespAssert` na classe **VCD**, é uma lista de objetos **Correspondence_Assertion**. Um objeto da classe **Correspondence_Assertion** é especificado por uma dupla <`vcdElem`, `odElem`>, que mapeia um conceito especificado na **VCD** a um conceito da Ontologia da Aplicação.

- **VCDManager**: constrói um objeto **VCD** a partir da especificação de uma VCD contida no arquivo XML de definição da VCD. O atributo `vcdDoc` é um objeto Document, correspondente a um documento XML contendo a especificação de uma VCD. Os métodos presentes nessa classe são:

  - `generateVCD`: constrói um objeto VCD a partir de um documento XML que contém sua especificação. Entrada: Arquivo XML; Saída: Objeto VCD.

  - `VCDtoFile`: operação inversa a anterior. Entrada: Objeto VCD; Saída: Arquivo XML que contém a especificação da VCD.

QueryGenerator é responsável pela geração da consulta SPARQL Q a partir da especificação parcial de um objeto VCD. O método `buildQuery` dessa classe recebe como entrada um objeto VCD, e sua saída é uma consulta SPARQL, correspondente à especificação dada.

6.4 Considerações Finais

Apresentamos neste capítulo um framework conceitual, o qual determina quais os módulos que devem ser implementados para possibilitar a execução do método proposto. Definimos cada módulo e artefato que deve ser utilizado neste processo, além de sugerir algumas ferramentas que podem desempenhar as atividades requeridas por cada módulo. Entretanto, a vantagem de definir um framework é que possibilitamos a determinação de quais os componentes básicos necessários para a execução deste método, ao mesmo tempo, garantindo ao desenvolvedor a liberdade de decidir quais ferramentas utilizar ou como desenvolvê-las, de forma que desempenhem os papéis de cada módulo satisfatoriamente. Apresentamos ainda o protótipo de uma ferramenta, que será capaz de instanciar o framework proposto, o que traz a vantagem de oferecer um suporte ao projetista da aplicação, na utilização deste método. Quanto ao estado de desenvolvimento desta ferramenta, encontra-se atualmente funcional um protótipo, que permite a atividade de criação de uma VCD, de acordo com a especificação $<S,P,Q,E>$. A implementação das demais funcionalidades poderá ser realizada em trabalhos subsequentes a este.

7. Discussão Final

7.1 Conclusões

Neste trabalho, propomos uma abordagem que visa minimizar os três principais problemas identificados, quanto ao desenvolvimento e utilização de Aplicações Web de Integração de Dados: (i) falta de um ciclo de desenvolvimento bem definido; (ii) dificuldade na especificação do conteúdo dinâmico e (iii) dificuldade na reutilização dos dados obtidos.

Para tanto, este trabalho intenciona ir além de uma simples ferramenta para projetar aplicações Web de integração de dados e mostra um método capaz de guiar alguém pela concepção de uma aplicação, de modo que não seja necessário usar linguagens ou softwares específicos. A abordagem proposta alivia os problemas identificados no projeto e desenvolvimento de tais aplicações, ao permitir uma especificação em alto nível do conteúdo dinâmico da aplicação sobre uma visão unificada dos conjuntos de dados consultados. Além disso, a abordagem ainda define como consultas e seus respectivos planos de execução podem ser gerados a partir da especificação dada.

Introduzimos o uso de Visões de Conteúdo Dinâmico para modelar conceitualmente os dados que são solicitados dinamicamente pelo usuário, durante suas interações com a aplicação. As VCDs são especificadas conceitualmente sobre uma Ontologia de Aplicação, que por sua vez modela o domínio específico de uma aplicação Web, funcionando com uma visão integrada de um conjunto de fontes a ser consultado. A vantagem de especificar as VCDs sobre a OA é que a especificação gerada é independente dos esquemas das fontes consultadas, de forma que mudanças nos esquemas das fontes não alteram a especificação conceitual das VCDs, senão somente os mapeamentos entre as fontes e a OA.

Podemos enumerar, pois, as contribuições desta dissertação como:

1. Definição e descrição de um método para o projeto de aplicações Web de integração de dados, que envolve três tarefas principais: especificação dos requisitos da aplicação sobre a Ontologia de Aplicação, onde tais requisitos são a especificação de quais dados devem ser apresentados como resultado; definição do plano de execução que calcula dinamicamente o conteúdo da aplicação Web; e anotação semântica do conteúdo obtido, seguida de sua publicação.

2. Especificação da etapa de Especificação Conceitual da Aplicação. Apresentamos a definição e descrição de uma infraestrutura baseada em ontologias para permitir a especificação conceitual do conteúdo da aplicação Web, em termos de visões virtuais parametrizadas, denominadas Visões de Conteúdo Dinâmico, definidas por uma quádrupla <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$>. Quanto às demais etapas do método (Definição dos Planos e Publicação dos Dados) foram citados outros trabalhos que podem eficientemente aplicados ou adaptados para a execução destas etapas;

3. Proposta de um ambiente que oferece suporte a cada passo das etapas de Integração Semântica, Especificação Conceitual, Definição dos Planos e Publicação de Dados, servindo de apoio à do método proposto. Tal ambiente é composto por um conjunto de módulos, que oferece suporte a cada etapa do método, bem como pela definição dos artefatos gerados e consumidos em cada uma destas etapas.

4. Projeto e prototipação de uma ferramenta que permita o gerenciamento de VCDs para uma aplicação Web de integração de dados.

7.2 Sugestões para Trabalhos Futuros

Como sugestões de trabalhos futuros a este destacam-se:

1. Desenvolver os elementos para suporte ao passo 2 do método (Definição da Aplicação)
    a. Desenvolver um ambiente para geração dos planos de execução das consultas;
    b. Estender o QEF (Query Execution Framework) para trabalhar com álgebra e operadores SPARQL, de forma que possa ser utilizado como ambiente de execução dos planos de execução;

2. Desenvolver os elementos para suporte ao passo 3 do método (Publicação de Dados)
    a. Definir um modelo para anotação semântica de VCDs que se utilize de sua especificação conceitual conforme definida neste trabalho;
    b. Desenvolver um serviço de *matchmaking* para realizar comparações entre os conceitos buscados pelo usuário e a descrição semântica de uma VCD, que deverá ser usado durante;

3. Realizar experimentos para testar o desempenho e a qualidade de uma aplicação Web de integração de dados desenvolvida através de ambiente funcional que ofereça suporte às três etapas do método.

.

8. Referências Bibliográficas

(Sendo padronizado, falta colocar aqui.)

9. Apêndice A – Estudo de Caso

Este estudo de caso foi construído para demonstrar o uso do método proposto neste trabalho, para o projeto de aplicações Web de integração de dados, na geração das VCDs para uma aplicação mashup denominada “Doenças & Drogas”, pertencente ao domínio da saúde, tratando dos aspectos de drogas relacionadas a determinadas doenças. Deseja-se saber dados detalhados (fórmula química, princípio ativo, efeitos colaterais, metabolismo etc.) sobre uma droga que trata uma dada doença.

A aplicação é composta por quatro páginas (Figura 4.1), que permitem a seguinte interação: o usuário pode procurar por uma doença, usando um critério definido (nome da doença), através da Home Page. O resultado desta consulta é mostrado na Página 1, que apresenta uma lista de nomes de doenças. Uma vez que o usuário seleciona um nome de doença, ele é direcionado para Página 2, que mostra uma lista de possíveis medicamentos para tratar a doença selecionada anteriormente. Finalmente, usuário seleciona um nome de medicamento (droga) e é redirecionado à Página 3, que mostra detalhes sobre as drogas selecionadas. A seguir, descrevemos os passos do método para especificação e implementação das VCDs da aplicação.

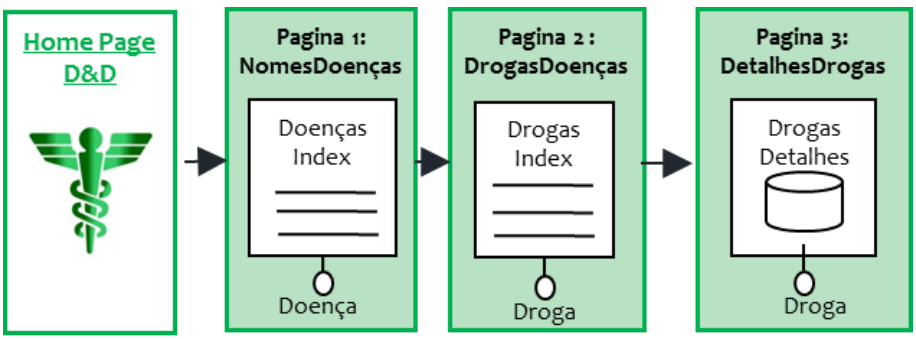
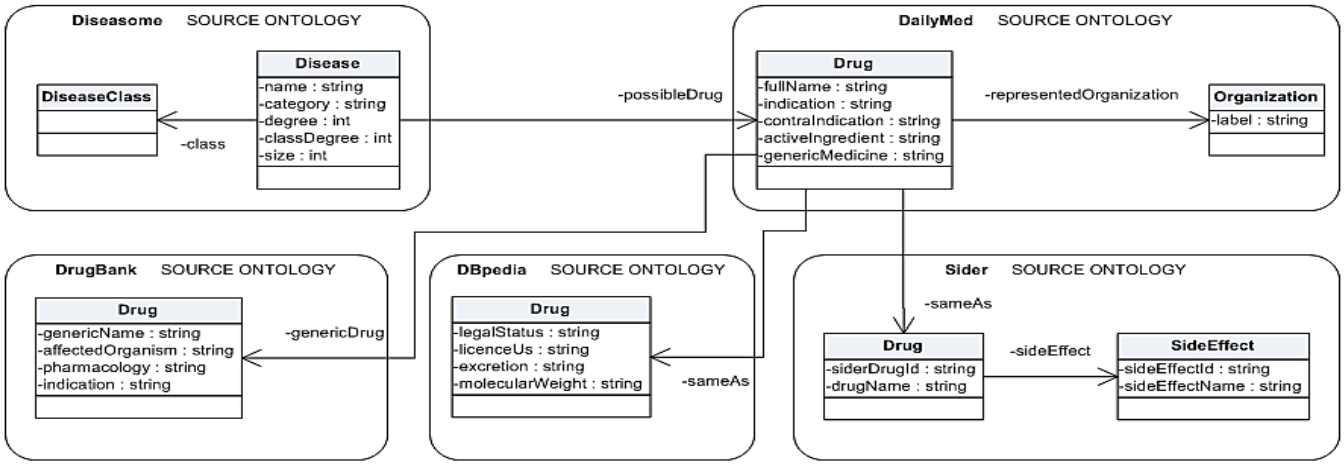


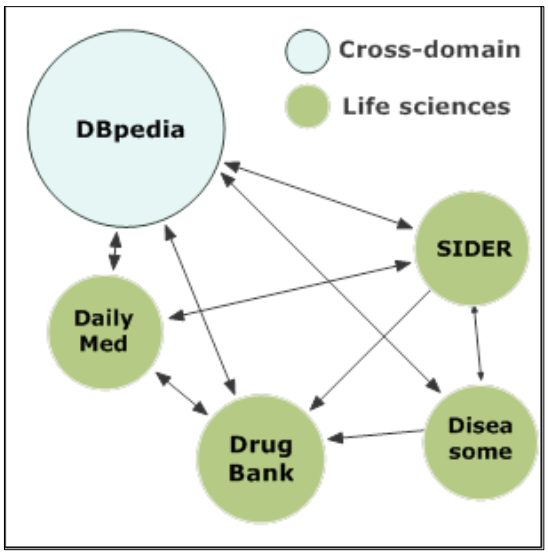
Figura 4.1 – Exemplo de aplicação Web de integração de dados: Mashup Doenças e Drogas

A.1 Integração Semântica

O primeiro passo da Integração semântica é definir uma Ontologia da Aplicação a qual representa o esquema conceitual integrado da mesma. Neste estudo de caso, esta ontologia baseia-se em ontologias existentes no Linked Data, para permitir que sua definição seja baseada em vocabulários padrão e tirar proveito infraestrutura existente no Linked Data. Para esta aplicação, as fontes relevantes são as ontologias Diseasome, DailyMed, DrugBank, DBpedia e Sider, as quais são registradas através dos URIs de *endpoints* SPARQL e podem ser selecionadas através da análise dos seus metadados, geralmente em formato voiD. As figuras a seguir mostram um esquema simplificado dos relacionamentos entre estas ontologias e uma visão destes relacionamentos dentro do Linked Data.



Relacionamentos das Ontologias Exportadas



Visão geral das OEs no Linked Data

Dada a identificação das informações relevantes para a aplicação, a Ontologia da Aplicação é uma visão unificada das fontes a serem consultadas, como pode ser visto na Figura 4.2.

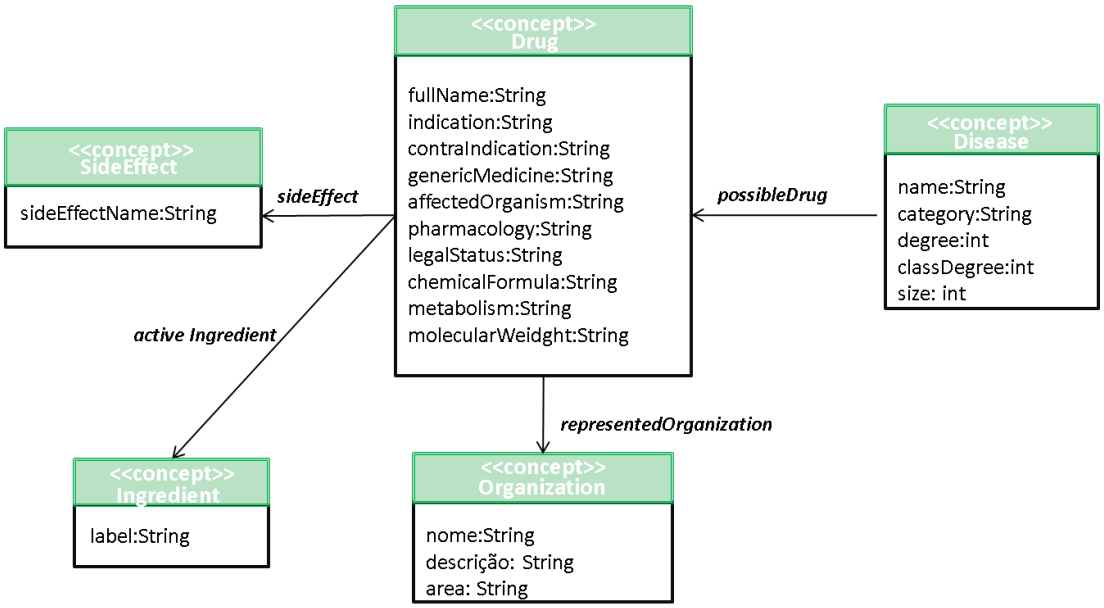


Figura 4.2 – Ontologia da Aplicação Mashup Doenças e Drogas

O segundo passo da Integração Semântica é a geração de Ontologias Exportadas, entretanto, como todas as fontes utilizadas já estão publicadas no Linked Data, o acesso a estas é realizado diretamente através de seus *endpoints* específicos. O terceiro passo desta etapa é a geração de mapeamentos entre a OA e as OEs. Neste caso, como as OEs já estão relacionadas entre si e a OA foi gerada a partir das partes relevantes das mesmas, um mapeamento R2R pode ser utilizado para relacioná-las. A seguir exibimos um modelo das regas de mapeamento entre uma OE e a OA e fornecemos uma amostra dos mapeamentos das classes da OA para as OEs.

```
oa:Drug(d) ⇐ dailymed:Drug(d)
oa:fullName(d, n) ⇐ dailymed:fullName(d, n), dailymed:Drug(d)
oa:Ingredient(i) ⇐ dailymed:Ingredient(i)
oa:label(i, l) ⇐ dailymed:label(i, l), dailymed:Ingredient(i)
```

Modelo de Mapeamentos

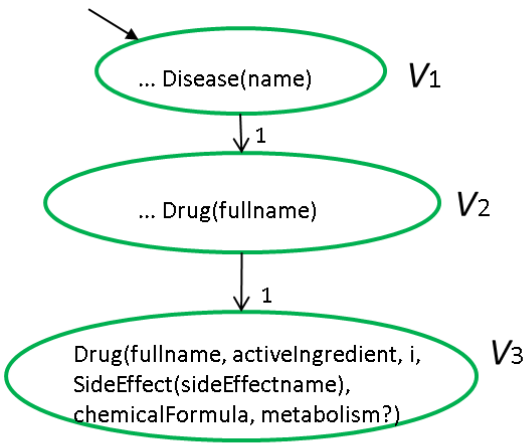
```
@prefix r2r: <http://www4.wiwiss.fu-berlin.de/bizer/r2r/> .
@prefix p: < http://www.lia.ufc.br/medical/resource/medical > .
@prefiz oa: oa: < http://www.lia.ufc.br/medical/DD.

# mapeamento das classes de DailyMed para a Ontologia da Aplicação
p:dailymedToOA
 a r2r:ClassMapping ;
 r2r:prefixDefinitions "dailymed: < http://www4.wiwiss.fu-
berlin.de/dailymed/sparql >" ;
 r2r:targetPattern "?s rdf:type oa:Drug" ;
 r2r:sourcePattern "?s rdf:type dailymed:Drug" .
 r2r:targetPattern "?s rdf:type oa:Organization" ;
 r2r:sourcePattern "?s rdf:type dailymed:Organization" .

# mapeamento das classes de DiseaseSome para a Ontologia da
Aplicação
p:diseasesomeToOA
 a r2r:ClassMapping ;
 r2r:prefixDefinitions "diseasesome:
<diseaseontology.sourceforge.net>" ;
 r2r:targetPattern "?s rdf:type oa:Disease" ;
 r2r:sourcePattern "?s rdf:type diseasesome:Disease" .

# mapeamento das classes de DrugBank para a Ontologia da Aplicação
p:drugbankToOA
 a r2r:ClassMapping ;
 r2r:prefixDefinitions "drugbank: < http://www4.wiwiss.fu-
berlin.de/drugbank >" ;
 r2r:targetPattern "?s rdf:type oa:Drug" ;
 r2r:sourcePattern "?s rdf:type drugbank:Drug" .

# mapeamento das classes de Sider para a Ontologia da Aplicação
p:siderToOA
 a r2r:ClassMapping ;
 r2r:prefixDefinitions "sider: <http://www4.wiwiss.fu-
berlin.de/sider/sparql>" ;
 r2r:targetPattern "?s rdf:type oa:Drug" ;
 r2r:sourcePattern "?s rdf:type sider:Drug".
r2r:targetPattern "?s rdf:type oa:SideEffect" ;
 r2r:sourcePattern "?s rdf:type sider:SideEffect"
```

Amostra dos mapeamentos de classes em R2R

A2. Especificação Conceitual da Aplicação

Nesta etapa serão modeladas as interações de uma aplicação e definidas as VCDs que definem o conteúdo dinâmico a ser recuperado em cada interação. O primeiro passo desta etapa é a definição das interações. No caso de nossa aplicação, o escopo de interações é reduzido, visto que o objetivo desta aplicação é ilustrar o método que apresentamos. Para a produção do UID que modela as interações, toma-se por base um levantamento de requisitos, em que as tarefas são contextualizadas para melhor compreensão das interações, como mostrado a seguir.

1. Identificação dos atores e tarefas:
    a. Ator: paciente clínico;
    b. Tarefa: Consultar drogas para uma doença
2. Especificação do Cenário:
    a. Um paciente clínico deseja saber informações sobre um medicamento (droga) receitado recentemente por seu médico. Para isto, ele deseja ver uma lista de nomes de doenças para escolher aquela que o interessa. Selecionado o nome de uma doença, a aplicação retorna uma lista com o nome completo das drogas normalmente utilizadas para tratar os sintomas da doença. O paciente então identifica e seleciona o nome do medicamento receitado por seu médico. A aplicação retorna então os detalhes da droga selecionada, apresentando seu nome completo, princípio ativo, efeitos colaterais, fórmula química e metabolismo, caso exista esta informação.
3. Especificação de um UID para cada tarefa a ser executada:



UID *drogasPorDoença*

O segundo e terceiro passo desta etapa é realizar a especificação das VCDs correspondentes para as interações do UID modelado (especificação parcial, seguida de geração automática da consulta). No UID drogasPorDoença há três interações, que resultam em três VCDs, especificadas na forma <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$>, exibidas a seguir.

VCD V1 *diseasesIndex*	
$\mathcal{S}$: **Attribute List**	*name: String*
$\mathcal{P}$: **Parameter List**	*none*
$\mathcal{Q}$: **Query**	*PREFIX md: <http://www.lia.ufc.br/medical/resource/medical> SELECT DISTINCT ?dsn WHERE { [] md:name ?dsn .} ORDER BY ?dsn*
$\mathcal{E}$: **Navigational Links**	*DrugsInfo: {Destiny: {drugsIndex}; Parameters:{name}}*

VCD V2 *drugsIndex*	
$\mathcal{S}$: **Attribute List**	*fullName: String*
$\mathcal{P}$: **Parameter List**	*name: String*
$\mathcal{Q}$: **Query**	*PREFIX md: <http://www.lia.ufc.br/medical/resource/medical> ?drName , ?x rdf:type oa:Disease AND ?x oa:name ?name AND ?y rdf:type oa:Drug AND ?y oa:fullName ?drName AND ?x oa:relatedTo ?y FILTER regex ?name,“s"
$\mathcal{E}$: **Navigational Links**	*DrugsMoreInfo: {Destiny: {drugsDetails}; Parameters:{fullName}}*

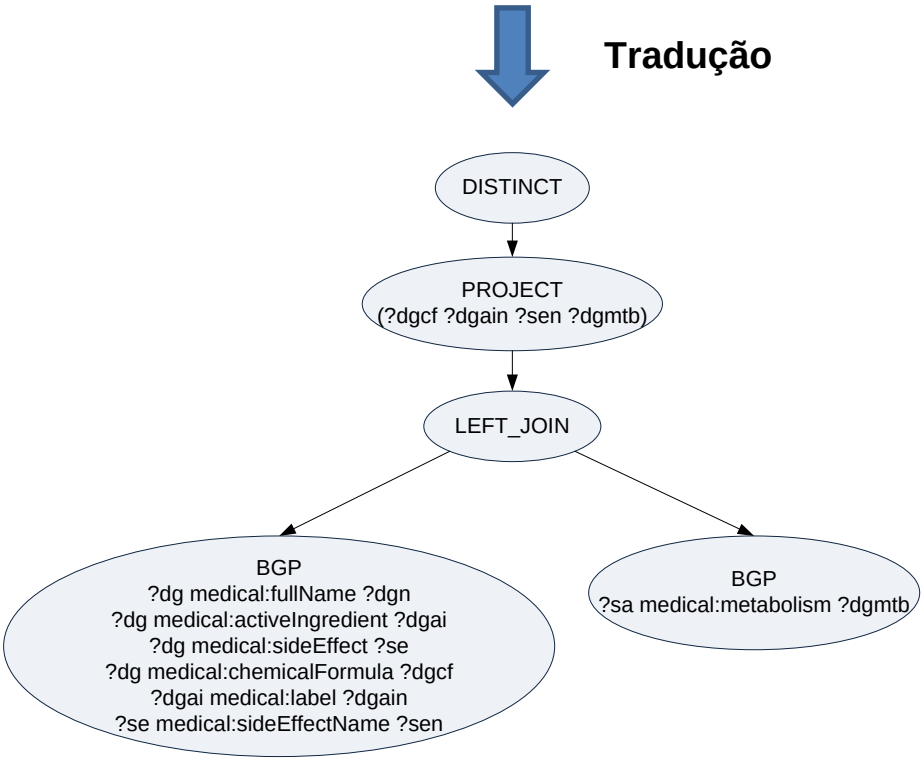
VCD V3 *drugsDetails*	
$\mathcal{S}$: **Attribute List**	*fullName: String* *activeIngredient: String* *indication: String* *sideEffectName: String* *legalStatus: String*
$\mathcal{P}$: **Parameter List**	*fullName: String*
$\mathcal{Q}$: **Query**	*PREFIX medical: <http://lia.ufc.br/med>* *SELECT DISTINCT ?dgcf ?dgain ?sen ?dgmtb* *WHERE {* *?dg medical:fullName ?dgn ;* *medical:activeIngredient ?dgai ;* *medical:sideEffect ?se ;* *medical:chemicalFormula ?dgcf .* *?dgai medical:label ?dgain .* *?se medical:sideEffectName ?sen .* *OPTIONAL {* *?dg medical:metabolism ?dgmtb .* *}*
$\mathcal{E}$: **Navigational Links**	*none*

A3. Definição dos Planos de Execução das VCDs

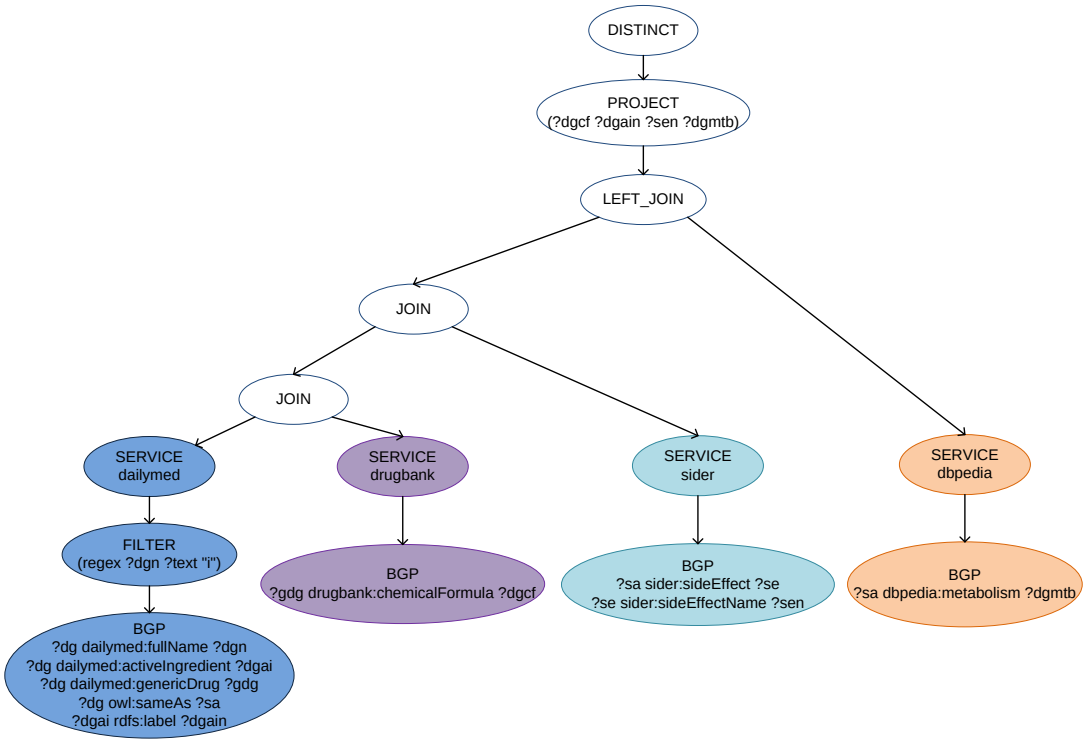
A terceira etapa do método proposto é a geração automática, com base na especificação conceitual, dos planos de execução das consultas que definem as VCDs. Visto que as consultas das VCDs V1 e V2 são bastante simples, apresentaremos para a VCD V3 todos os passos para a geração do plano de execução final. A geração do plano de consulta federado ocorre em tempo de projeto em quatro passos, baseado em [Pinheiro et al. 2009]:

1. Tradução - Conversão da consulta SPARQL para plano de consulta em álgebra SPARQL sobre a OA.

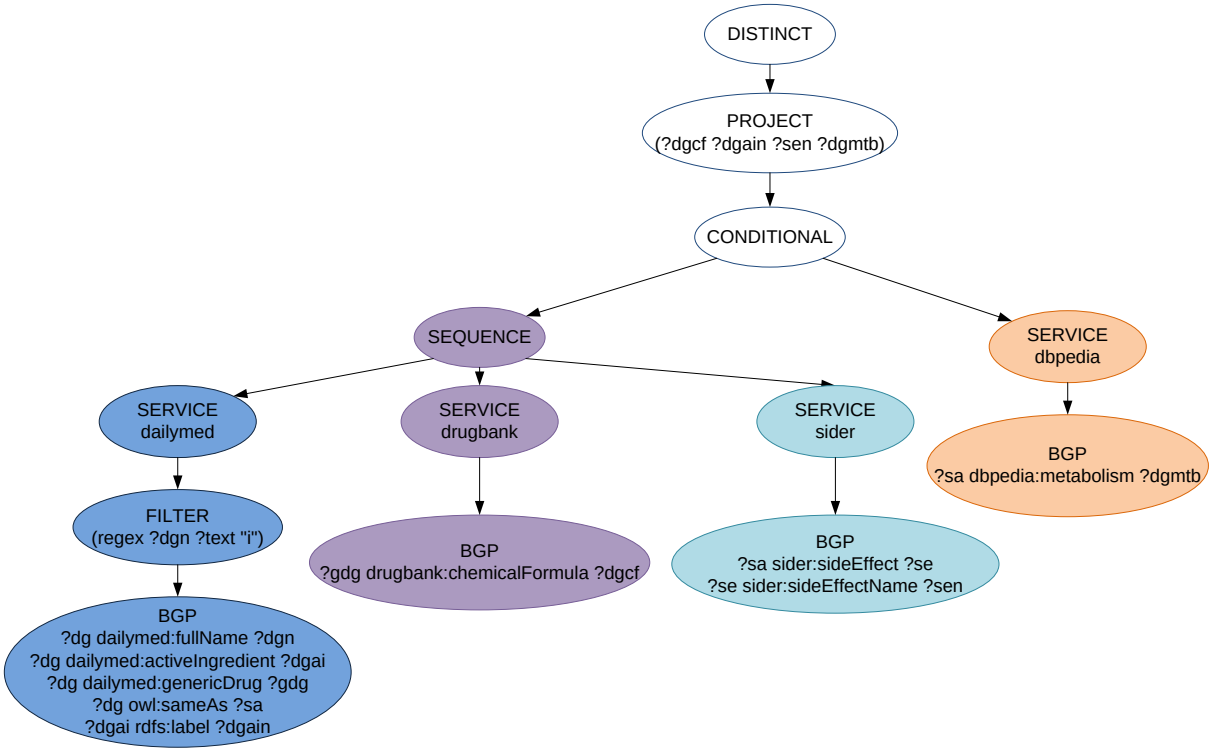
```
PREFIX medical: <http://lia.ufc.br/med>
SELECT DISTINCT ?dgcf ?dgain ?sen ?dgmtb
WHERE {
   ?dg medical:fullName ?dgn ;
       medical:activeIngredient ?dgai ;
       medical:sideEffect ?se ;
       medical:chemicalFormula  ?dgcf .
   ?dgai medical:label ?dgain .
   ?se medical:sideEffectName ?sen .
  OPTIONAL {
    ?dg medical:metabolism ?dgmtb .
  }
```



2. Reformulação - Transformação do plano algébrico para plano de consulta federado sobre as OEs.



3. Otimização – otimização do plano de consulta federado sobre as OEs, para obter melhor desempenho.



4. Geração do Plano de Execução – Uma vez gerado o plano otimizado, são definidas as operações para realizar as subconsultas sobre as fontes e construir o resultado final, gerando um plano final composto de operadores para acesso e controle do fluxo de dados.

A4. Publicação dos Dados

Consideramos que serviços de acesso a dados dão acesso às VCDs e estão registrados em catálogos de serviços Web. Para manter comparáveis as diversas descrições de VCDs contidas em um ou mais catálogos, é crucial que tais descrições estejam baseadas em um vocabulário compartilhado entre as diversas fontes, neste caso, a Ontologia da Aplicação. Portanto, a descrição do serviço em si é dada pela tripla <$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$> em termos da OA, possibilitando encontra-la através da busca em um catálogo. Neste trabalho não foi definido um formalismo ou padrão para a representação da anotação semântica da VCD em termos da tripla<$\mathcal{S}$, $\mathcal{P}$, $\mathcal{Q}$>, pelo que esta parte do estudo de caso foi apenas idealizada, segundo descrito no Capítulo 5.

10. Apêndice B – Algoritmo para Geração de Consultas

O algoritmo GeraConsulta é responsável por, dada a especificação parcial de uma VCD, gerar a consulta Q que define uma VCD sobre a OA. GeraConsulta recebe como entrada o esquema XML ($\mathcal{S}$), a lista de parâmetros ($\mathcal{P}$) e os mapeamentos ($\mathcal{M}$) de uma VCD (caso tenham sido definidos) e a Ontologia da Aplicação e retorna a consulta Q que permite a definição da VCD sobre a Ontologia da Aplicação.

GeraConsulta(VCD,OA)

Entrada: um objeto VCD <S, P, E> , a Ontologia de Aplicação: appOntology e um conjunto M de assertivas, caso tenham sido definidas. Caso contrário, M é por padrão uma correspondência nominal entre os termos da VCD e OA.

Saída: uma consulta SPARQL Qs

```
1.  //Geração do cabeçalho da consulta
2.      Qs +=  "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
3.      $uri = obterURI(appOntology);
4.          "PREFIX oa:" + $uri;
5.  //Geração da cláusula FROM
6.      Qs += Qs + "FROM"+$uri;
7.  //Geração da Cláusula SELECT
8.      Qs = Qs + "SELECT"
9.      Para cada $atributo em S  faça
10.         Qs += "?"+$atributo+"_"+ obterDominio($atributo)+"  ";
11.     Fim Para
12.     Para cada $atributo opcional de S  faça
13.         Qs +="?"+$atributo+"_"+ obterDomínio($atributo)+"  "
14.     Fim Para
```

```
15.   //Geração da Cláusula WHERE
16.       Qs = Qs+WHERE+”{“
17.       Para cada $classeDeAtributo em M  faça
18.           Qs = Qs+”?”$classeDeAtributo+" rdf:type oa:"+$classeDeAtributo+";"
19.           $atributosClasse = obterAtributos(VCD,$classeDeAtributo );
              $atributosOpcionaisClasse = obterOpcAtributos(VCD,$classeDeAtributos);
20.           Para cada $atributo de $atributosClasse faça
21.               Qs += " oa:"+$atributo+"  ?"+$atributo+"_"+$classe+";"
22.           Fim Para
23.           Para cada $atributo de $atributosOpcionaisClasse faça
24.               Qs+="OPTIONAL”+"{"+" ?"+$classe+"
25.               oa:"+atributoOpcional +" ?"+atributoOpcional+"_"+$classe+"}.");
26.           Fim Para
27.           Para cada $parametro em P  faça
28.               $parametroPropDados = $parametro.obterPropriedadeDados();
29.               $parametroOperador =  $parametro.obterOperador();
30.               $parametroValor =  $parametro.obterValor();
31.               Se $parametroPropDados possui range do tipo String em appOntology
                  Então
32.                   Q+=“FILTER”+"  REGEX” +" (
33.                   ?"+parametroPropDados+"_"+$classe +" ,"+$parametroValor+”).");
34.               Senão
35.                   Q+=“FILTER"+”(?"+parametroPropDados+"
36.                   ?"+parametroPropDados+"_"+$classe +", "+$parametroValor+”).");
37.               Fim Se
38.           Fim Para
39.           Para cada $classeFilha de $classeDeAtributo em M  faça
40.              $propObjeto = obterPropObjeto($classeDeAtributo, $classeFilha)
41.              Qs = Qs +” ”+$propObjeto+” ”+$classeFilha+”;”
42.           Fim Para
43.           Qs+= “.;”
44.       Fim Para
45.   //Geração da Cláusula ORDER BY
46.       Se existe condição de ordenação em S então
47.           Qs+=”ORDER BY”
48.           Para cada $atributo dos atributos de ordenação em S  faça
49.                Qs+=”?”+$atributo+”_”+ obterDominio($atributo)+””;
```

50. **Fim Para**
51. **Fim Se**
52. Qs+=".}";
53. **Retorne** Qs;
Nota: (i) obterDominio é uma função que recebe como entrada uma propriedade de dados e retorna a classe da ontologia que corresponde ao domínio desta propriedade. (ii) obterAtributos é uma função que recebe como entrada um objeto VCD e uma classe da ontologia e retorna as propriedades de dados que possuem como domínio a classe dada. A propriedade retornada é atributo da VCD. (iii) obterOpcAtributos é uma função que recebe como entrada um objeto VCD e uma classe da ontologia e retorna as propriedades de dados opcionais à consulta que possuem como domínio a classe dada. A propriedade retornada é atributo opcional da VCD. (iv) obterPropObjeto é uma função que recebe como entrada a classe domain e a classe range, e retorna a propriedade de objeto que as relacionam na ontologia. (v) obterParametros é uma função que recebe como entrada um objeto VCD e uma classe da ontologia e retorna os parâmetros da VCD. Cada parâmetro retornado é correspondente a uma propriedade de dados, cujo domínio é a classe dada.

Primeiramente é gerado o cabeçalho da consulta com a cláusula PREFIX, contendo os prefixos que serão utilizados, sendo o prefixo rdf fixo e o prefixo da ontologia da aplicação obtido através da função obterURI. A cláusula FROM aponta para a ontologia da aplicação. A cláusula SELECT gerada indica os atributos obrigatórios e opcionais, contidos no esquema $\mathcal{S}$, que deverão ser exibidos como resultado da consulta. Na cláusula WHERE, são montadas as triplas do padrão da consulta, a partir das correspondências definidas no mapeamento $\mathcal{M}$, entre atributos do esquema e propriedades de dados da ontologia. A cláusula FILTER pode ser definida com ou sem o modificador REGEX (para análise de strings), com base nos parâmetros definidos na lista de parâmetros $\mathcal{P}$. Por fim, a cláusula ORDER BY é definida, caso haja atributos de ordenação definidos em $\mathcal{S}$.

As funções para obtenção de domínio, URI, atributos, propriedades de dados e de objeto e parâmetros utilizam funções nativas da API Jena[28] para manipulação de ontologias. A principal limitação deste algoritmo é a impossibilidade de gerar consultas cíclicas, visto que um caminho (*pattern tree*) é traçado desde a classe base até a classe que se deseja consultar, sem a possibilidade de autoreferência.

[28] incubator.apache.org.br/jena

11. Apêndice C – Projeto da Ferramenta

Cada etapa do método foi concebida como um subsistema ou camada de uma ferramenta, segundo a divisão e os módulos propostos no framework conceitual. Dessa forma, descrevemos quais são as atividades pertinentes a cada etapa e quais os atores que as desempenharão no ambiente proposto, através da descrição de casos de uso.

C1. Funcionalidades da Camada de Integração Semântica

O ator que interage com a ferramenta, para a execução das atividades de Integração Semântica, é o Especialista de Domínio, o qual representa uma pessoa ou grupo, que possui a experiência ou conhecimento a serem utilizados na definição da semântica e do esquema conceitual da aplicação. Apresentamos a seguir o detalhamento dos casos de uso referentes à etapa de Integração Semântica.

Caso de Uso 1: Selecionar/Editar OA

Descrição do caso de uso: Este caso de uso permite que o Especialista do Domínio selecione ou edite uma Ontologia de Aplicação, correspondente ao Esquema Conceitual da Aplicação.

Entradas e pré-condições: não possui

Saídas e pós-condição: Uma Ontologia RDF é carregada na aplicação e persistida no Repositório de Artefatos. Um aviso de status da operação é exibido na tela.

Fluxo principal (Seleção)

1. O usuário seleciona a aba [Integração Semântica];
2. O usuário clica em [Selecionar Ontologia de Aplicação];

3. O sistema exibe uma caixa de diálogo para seleção de arquivo;
4. O usuário indica o caminho do arquivo RDF da ontologia;
5. O usuário clica em [Carregar Ontologia];
6. O sistema verifica se o arquivo é válido;
7. O sistema carrega a ontologia;
8. O sistema persiste a ontologia no diretório <Ontologias de Aplicação>, do Repositório de Artefatos;
9. O sistema mostra um aviso de status da operação;

Fluxo principal (Edição)

1. O usuário seleciona a aba [Integração Semântica];
2. O usuário clica em [Selecionar Ontologia de Aplicação];
3. O sistema exibe uma caixa de diálogo para seleção de arquivo;
4. O usuário indica o caminho do arquivo RDF da ontologia;
5. O usuário clica em [Carregar Ontologia];
6. O sistema verifica se o arquivo é válido;
7. O sistema carrega a ontologia;
8. O sistema abre a Ontologia em modo Edição através das funcionalidades do Protégé.
9. O usuário realiza as edições desejadas.
10. O sistema persiste a ontologia no diretório <Ontologias de Aplicação>, do Repositório de Artefatos;
11. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA 003] Arquivo Inválido

[FA 004] Erro no carregamento da ontologia

Caso de Uso 2: Gerar OEs

Descrição do caso de uso: Este caso de uso permite que o Especialista do Domínio gere as Ontologias Exportadas de cada fonte local.

Entradas e pré-condições: O usuário haver carregado a OA e, pelo menos, uma OL (gerada diretamente a partir da fonte a ser consultada).

Saídas e pós-condição: Uma Ontologia RDF em termos da OA é carregada na aplicação e persistida no Repositório de Artefatos. Um aviso de status da operação é exibido na tela.

Fluxo principal

1. O usuário seleciona a aba [Integração Semântica];
2. O usuário clica em [Gerar OEs];
3. O sistema exibe as OLs existentes;
4. O usuário seleciona a(s) OL(s) a partir das quais deseja obter OEs;

5. O sistema gera o conjunto de OEs correspondente;
6. O sistema persiste a ontologia no diretório <Ontologias Exportadas>, do Repositório de Artefatos;
7. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[[FA 002] Erro na geração da ontologia.

Caso de Uso 3: Definir Mapeamentos

Descrição do caso de uso: Este caso de uso permite que o Especialista do Domínio defina as correspondências entre os conceitos da Ontologia de Aplicação e os conceitos das Ontologias Exportadas

Entradas e pré-condições: O usuário haver carregado a OA e, pelo menos, uma OE.

Saídas e pós-condição: Um arquivo de mapeamentos é gerado e persistido no Repositório de Artefatos. O usuário deverá validar manualmente os mapeamentos obtidos (pós-matching).

Fluxo principal

1. O usuário seleciona a aba [Integração Semântica];
2. O usuário clica em [Definir Mapeamentos];
3. O sistema apresenta na tela a OA e as OEs disponíveis;
4. O usuário seleciona a(s) OE(s) que deseja mapear com a OA;
5. O usuário clica em [Gerar Mapeamentos];
6. O sistema gera os mapeamentos automaticamente;
7. O sistema persiste o arquivo de mapeamentos no diretório <Mapeamentos>, do Repositório de Artefatos;
8. O sistema mostra um aviso de status da operação;
9. O sistema exibe na tela os mapeamentos obtidos;
10. Caso os mapeamentos obtidos não sejam suficientes e:

    a. O usuário deseje alterar uma regra de mapeamento:
        1. O usuário clica em [Editar Mapeamentos];
        2. O sistema exibe a tela de edição de mapeamentos;
        3. O usuário seleciona a regra de mapeamento que deseja alterar;
        4. O sistema habilita uma caixa de texto, contendo a regra selecionada;
        5. O usuário altera a regra;
        6. O usuário clica em [Salvar alterações];
        7. O sistema checa a validade da regra;
        8. O sistema realiza as alterações no arquivo de mapeamentos;

9. Retorna para o Passo 7 do fluxo principal deste caso de uso.

b. O usuário deseje acrescentar uma nova regra de mapeamento:
    1. O usuário clica em [Editar Mapeamentos];
    2. O sistema exibe a tela de edição de mapeamentos;
    3. O usuário clica em [Adicionar Regra];
    4. O sistema habilita uma caixa de texto;
    5. O usuário digita a nova regra;
    6. O usuário clica em [Salvar alterações];
    7. O sistema checa a validade da nova regra;
    8. O sistema insere a nova regra no arquivo de mapeamentos;
    9. Retorna para o Passo 7 do fluxo principal deste caso de uso.

c. O usuário deseja excluir uma regra de mapeamento:
    1. O usuário clica em [Editar Mapeamentos];
    2. O sistema exibe a tela de edição de mapeamentos;
    3. O usuário seleciona a regra de mapeamento que deseja excluir;
    4. O usuário clica em [Excluir regra];
    5. O usuário clica em [Salvar alterações];
    6. O sistema exclui a regra do arquivo de mapeamentos;
    7. Retorna para o Passo 7 do fluxo principal deste caso de uso.

11. Caso os mapeamentos obtidos sejam suficientes:
    1. O usuário clica em [Confirmar Mapeamentos];

Fluxos alternativos

[FA 003] Arquivo Inválido

[FA 004] Erro no carregamento da ontologia

[FA005] Regra de mapeamento inválida

C.2 Funcionalidades da Camada de Especificação Conceitual da Aplicação

O ator que interage com a ferramenta, para a execução das atividades de Especificação Conceitual da Aplicação, é o Projetista da Aplicação, o qual representa uma pessoa que possui conhecimentos sobre as regras de negócio da aplicação.

Caso de Uso 1: Modelar Interações

Descrição do caso de uso: Este caso de uso permite que o Projetista da Aplicação modele os UIDs da aplicação.

Entradas e pré-condições: não há.

Saídas e pós-condição: Um arquivo contendo a modelagem dos UIDs é gerado e persistido no Repositório de Artefatos.

Fluxo principal

1. O usuário seleciona a aba [Especificação Conceitual da VCD];
2. O usuário clica em [Novo UID];
3. O sistema exibe um conjunto de ferramentas para o desenho gráfico do UID.
4. O usuário desenha e preenche as propriedades do(s) UID(s).
5. O usuário clica em [Salvar Modelagem];
6. O sistema persiste o arquivo da modelagem no diretório <UIDs>, do Repositório de Artefatos;
7. O sistema mostra um aviso de status da operação;

Fluxo Alternativo

[FA011] – Erro ao salvar arquivo.

Caso de Uso 2: Criar VCD

Descrição do caso de uso: Este caso de uso permite que o Projetista da Aplicação especifique uma nova VCD.

Entradas e pré-condições: O usuário haver carregado uma OA e haver modelado, pelo menos, um UID.

Saídas e pós-condição: Um arquivo XML, que representa a especificação conceitual da VCD é gerado e persistido no Repositório de Artefatos.

Fluxo principal

1. O usuário seleciona a aba [Especificação Conceitual da VCD];
2. O usuário clica em [Nova VCD];
3. O sistema exibe os UIDs modelados;
4. O usuário seleciona a interação do UID para a qual deseja projetar uma VCD;
5. Os Casos de Uso de número 2.1, 2.2, 2.3 (optativamente), 2.4, e 2.5 são executados, nesta ordem;
6. O sistema exibe a VCD graficamente, com os parâmetros preenchidos, na tela da aplicação;
7. O usuário clica em [Salvar VCD];
8. O sistema solicita que o usuário atribua um nome para o arquivo;
9. O usuário digita o nome e clica em [Ok];
10. O sistema persiste o arquivo XML da VCD no diretório <VCDs>, do Repositório de Artefatos;
11. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA009] – Não há OA carregada

Caso de Uso 2.1: Definir esquema XML

Descrição do caso de uso: Este caso de uso permite a definição do Esquema XML, que define os atributos de uma VCD.

Entradas e pré-condições: Ocorrer durante o Caso de Uso de Criação ou Edição da VCD.

Saídas e pós-condição: O sistema retorna para a tela de exibição gráfica da VCD, contendo o campo [Esquema] preenchido.

Fluxo principal

1. O usuário clica em [Esquema], na representação gráfica da VCD;
2. O usuário clica em [Selecionar Esquema];
3. O sistema exibe uma caixa de diálogo para seleção de arquivo;
4. O usuário indica o caminho do arquivo XML de esquema;
5. O usuário clica em [Carregar Esquema];
6. O sistema verifica se o arquivo é válido;
7. O sistema carrega o esquema;
8. O sistema mostra um aviso de status da operação;
9. O sistema retorna para a tela de exibição gráfica da VCD;

Fluxos alternativos

[FA003] Arquivo Inválido

Caso de Uso 2.2: Definir parâmetros

Descrição do caso de uso: Este caso de uso permite a definição dos parâmetros utilizados por uma VCD.

Entradas e pré-condições: Ocorrer durante o Caso de Uso de Criação ou Edição da VCD.

Saídas e pós-condição: O sistema retorna para a tela de exibição gráfica da VCD, contendo o campo [Parâmetros] preenchido.

Fluxo principal

1. O usuário clica em [Parâmetros], na representação gráfica da VCD;
2. O sistema exibe um conjunto de possíveis parâmetros (com base na definição do UID) e seus tipos;
3. O usuário seleciona os parâmetros que deseja utilizar;
4. O usuário clica em [Ok];
5. O sistema mostra um aviso de status da operação;
6. O sistema retorna para a tela de exibição gráfica da VCD;

Fluxos alternativos

[FA006] – Nenhum parâmetro selecionado

Caso de Uso 2.3: Gerar assertivas de correspondência

Descrição do caso de uso: Este caso de uso permite a definição gráfica de assertivas de correspondência, entre o esquema XML da VCD e a Ontologia de Domínio, caso seja necessário.

Entradas e pré-condições: Ocorrer durante o Caso de Uso de Criação ou Edição da VCD. O usuário haver carregado o Esquema da VCD.

Saídas e pós-condição: O sistema retorna para a tela de exibição gráfica da VCD, contendo o campo [Correspondências] preenchido.

Fluxo principal

1. O usuário clica em [Correspondências], na representação gráfica da VCD;
2. O sistema exibe a tela de definição de correspondências, contendo, do lado esquerdo, o Esquema XML em forma de árvore, detalhando seus atributos e, do lado direito, a Ontologia de Domínio em forma de grafo, detalhando suas propriedades de dados e objetos;
3. Para cada atributo do esquema, o usuário seleciona uma propriedade da ontologia;

4. O usuário clica em [Ok];
5. O sistema mostra um aviso de status da operação;
6. O sistema retorna para a tela de exibição gráfica da VCD;

Fluxos alternativos

[FA007] – Definição de correspondências incompleta.

Caso de Uso 2.4: Gerar consulta SPARQL

Descrição do caso de uso: Este caso de uso permite a geração automática da consulta SPARQL que define a VCD sobre a OA.

Entradas e pré-condições: Ocorrer durante o Caso de Uso de Criação ou Edição da VCD. O usuário haver definido parâmetros e gerado as assertivas de correspondência.

Saídas e pós-condição: O sistema retorna para a tela de exibição gráfica da VCD, contendo o campo [Consulta] preenchido.

Fluxo principal

1. O usuário clica em [Consulta], na representação gráfica da VCD;
2. O sistema verifica se há parâmetros selecionados e as correspondências geradas e então habilita o botão [Gerar Consulta];
3. O usuário clica em [Gerar Consulta];
4. O sistema gera a consulta SPARQL correspondente à especificação;
5. O sistema mostra um aviso de status da operação;
6. O sistema retorna para a tela de exibição gráfica da VCD;

Fluxos alternativos

[FA006] – Nenhum parâmetro definido.

[FA007] – Definição de correspondências incompleta.

Caso de Uso 2.5: Definir elos

Descrição do caso de uso: Este caso de uso permite a definição dos elos navegacionais entre as VCDs.

Entradas e pré-condições: Ocorrer durante o Caso de Uso de Criação ou Edição da VCD. O usuário haver definido parâmetros.

Saídas e pós-condição: O sistema retorna para a tela de exibição gráfica da VCD, contendo o campo [Elos] preenchido. Os elos representarão relacionamentos entre as VCDs, no modo de exibição.

Fluxo principal

1. O usuário clica em [Elos], na representação gráfica da VCD;
2. O sistema verifica se há parâmetros selecionados e habilita o botão[Definir elos];
3. O usuário clica em [Definir elos];
4. O sistema exibe uma caixa de texto, contendo sugestões de elos, com base na definição do UID;
5. O usuário seleciona os elos desejados;
6. O usuário clica em [Ok];
7. O sistema mostra um aviso de status da operação;
8. O sistema retorna para a tela de exibição gráfica da VCD;

Fluxos alternativos

[FA006] – Nenhum parâmetro definido.

Caso de Uso 10: Editar VCD

Descrição do caso de uso: Este caso de uso permite que o Projetista da Aplicação edite uma VCD já criada.

Entradas e pré-condições: O usuário ter criado, pelo menos, uma VCD.

Saídas e pós-condição: Um arquivo XML, que representa a VCD é alterado e persistido no Repositório de Artefatos.

Fluxo principal

1. O usuário seleciona a aba [Especificação Conceitual da VCD];
2. O usuário clica em [Editar VCD];
3. O sistema exibe a representação gráfica das VCDs existentes, bem como seus relacionamentos;
4. O usuário seleciona a VCD que deseja editar;
5. O sistema exibe a representação gráfica detalhada da VCD selecionada;
6. O usuário seleciona o campo da VCD que deseja alterar:
    a. Caso a alteração seja no campo Esquema, o Caso de Uso número 5 é executado;
    b. Caso a alteração seja no campo Parâmetros, o Caso de uso número 6 é executado;
    c. Caso a alteração seja no campo Correspondências, o Caso de uso número 7 é executado;
    d. Caso a alteração seja no campo Consulta, o Caso de uso número 8 é executado;
    e. Caso a alteração seja no campo Elos, o Caso de uso número 9 é executado;
7. O sistema exibe a VCD graficamente, com os parâmetros preenchidos, na tela da aplicação;
8. O usuário clica em [Salvar VCD];

9. O sistema solicita que o usuário atribua um nome para o arquivo;
10. O usuário digita o nome e clica em [Ok];
11. O sistema persiste o arquivo XML da VCD no diretório <VCDs>, do Repositório de Artefatos;
12. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA008] – Não há VCDs criadas

Caso de Uso 11: Excluir VCD

Descrição do caso de uso: Este caso de uso permite que o Projetista da Aplicação exclua uma VCD já criada.

Entradas e pré-condições: O usuário ter criado, pelo menos, uma VCD.

Saídas e pós-condição: O arquivo XML que representa a VCD é excluído do Repositório de Artefatos.

Fluxo principal

1. O usuário seleciona a aba [Especificação Conceitual da VCD];
2. O usuário clica em [Excluir VCD];
3. O sistema exibe a representação gráfica das VCDs existentes, bem como seus relacionamentos;
4. O usuário seleciona a VCD que deseja excluir;
5. O sistema exibe uma mensagem de confirmação;
6. O usuário clica em [Ok];
7. O sistema exclui o arquivo XML da VCD no diretório <VCDs>, do Repositório de Artefatos;
8. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA008] – Não há VCDs criadas

C.3 Funcionalidades da Camada de Implementação da VCD

O ator que interage com a ferramenta, para a execução das atividades de Implementação da VCD (relativa a Definição da Aplicação e Publicação dos Dados), é o projetista da aplicação, assim como na fase de especificação da VCD.

Caso de Uso 1: Gerar Plano de Execução

Descrição do caso de uso: Este caso de uso permite que um Plano de Execução seja gerado automaticamente, para a implementação de uma VCD, seguindo o processo interno definido na etapa de Definição da Aplicação.

Entradas e pré-condições: O usuário ter carregado uma OA e ter criado, pelo menos, uma VCD.

Saídas e pós-condição: Um plano de consulta em SPARQL é persistido no Repositório de Artefatos. Um aviso de status da operação é exibido na tela.

Fluxo principal

1. O usuário seleciona a aba [Implementação da VCD];
2. O usuário clica em [Gerar Plano];
3. O sistema exibe a representação gráfica das VCDs existentes, bem como seus relacionamentos;
4. O usuário seleciona as VCDs para as quais deseja gerar um plano de execução;
5. O sistema gera o plano de consulta mais eficiente para cada consulta;
6. O sistema solicita que o usuário atribua um nome para o arquivo;
7. O usuário digita o nome e clica em [Ok];
8. O sistema persiste os planos de consulta no diretório <Planos de Consulta> do Repositório de Artefatos;
9. Repositório de Artefatos;
10. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA008] – Não há VCDs criadas

[FA009] – Não há OA carregada

Caso de Uso 2: Anotar SWID Semanticamente

Descrição do caso de uso: Este caso de uso permite que o projetista da aplicação realize a anotação semântica das VCDs.

Entradas e pré-condições: O usuário ter criado, pelo menos, uma VCD e carregado uma OA.

Saídas e pós-condição: Um arquivo WSDL anotado é gerado e persistido no Repositório de Artefatos. Um aviso de status da operação é exibido na tela.

Fluxo principal

1. O usuário seleciona a aba [Implementação da VCD];
2. O usuário clica em [Anotar VCD];
3. O sistema exibe uma lista de VCDs disponíveis;
4. O usuário seleciona a VCD que desejar anotar;
5. O sistema exibe a tela de anotação semântica, contendo, do lado esquerdo, as opções de anotação, e do lado direito, os elementos de descrição do arquivo WSDL.
6. Para cada elemento, o usuário seleciona uma entidade na OA que o descreva;
7. O usuário clica em [Salvar Anotação];
8. O sistema solicita que o usuário atribua um nome para o arquivo;
9. O usuário digita o nome e clica em [Ok];
10. O sistema persiste o arquivo de anotação no diretório <Anotações Semânticas>, do Repositório de Artefatos;
11. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA009] – Não há OA carregada

[FA010] – Não há VCD criada

Caso de Uso 3: Publicar VCD

Descrição do caso de uso: Este caso de uso permite que o projetista da aplicação realize a publicação das VCDs anotadas em um repositório Web.

Entradas e pré-condições: O usuário ter anotado, pelo menos, uma VCD e haver uma conexão estabelecida com a Web.

Saídas e pós-condição: A VCD é registrada em um repositório UDDI. Um aviso de status da operação é exibido na tela.

Fluxo principal

1. O usuário seleciona a aba [Implementação da VCD];
2. O usuário clica em [Publicar VCD];
3. O sistema exibe uma lista de VCDs disponíveis;
4. O usuário seleciona a VCD que desejar publicar;
5. O sistema exibe a lista dos repositórios UDDI conhecidos.
6. O usuário seleciona o repositório em que deseja publicar seu serviço;
7. O usuário clica em [Publicar];
8. O sistema realiza a conexão com o repositório selecionado e registra o serviço escolhido;
9. O sistema mostra um aviso de status da operação;

Fluxos alternativos

[FA012] – Não há conexão com a Internet.

12. Apêndice D - Levantamento de Tecnologias

Após a identificação das funcionalidades que a ferramenta deve contemplar, realizamos o levantamento de algumas das tecnologias que podem ser utilizadas para a implementação destas funcionalidades, ou ferramentas e plug-ins que possam ser reutilizados ou adaptados para o mesmo fim. O quadro a seguir apresenta o resultado do levantamento realizado.

Camada de Integração Semântica	
Funcionalidade	**Tecnologia**
Obter ontologia a partir de uma fonte local	• RDB2Onto [Trin et al. 2006] • RELATIONAL.OWL [Laborda and Conrad 2006] • D2RQ [Bizer and Seaborne 2004] .
Selecionar ou Editar OA	• Protégé • KAON • DOME • OilEd
Gerar OE	• Ferramenta que implementa a abordagem proposta por [Sacramento and Vidal 2010], a qual ainda encontra-se em desenvolvimento.
Gerar Mapeamentos	• Ferramenta que implementa a abordagem proposta por [Sacramento and Vidal 2010], a qual ainda encontra-se em desenvolvimento

Camada de Especificação Conceitual da **Aplicação**	
Funcionalidade	**Tecnologia**
Modelagem Conceitual	• Ferramenta para diagramação de UIDs [Vilain et al 2000] • UMLGraph [Spinellis 2004]
Criar VCD / Editar VCD / Excluir VCD	• Manipulação XML: API SAX, JDOM • Manipulação de ontologias: Jena, JRDF, OWL API • Manipulação SPARQL: Jena, SPARQL Engine for Java • Representação Gráfica: UML Graph

Camada de Implementação da VCD	
Funcionalidade	**Tecnologia**
Gerar plano de execução	• Ferramenta que implementa a abordagem proposta por [Pinheiro et al 2010], a qual ainda encontra-se em desenvolvimento
Anotar VCD	• Radiant, SAWSDL Plugin, SAWSDL4J
Publicar VCD	• JAXR for UDDI, jUDDI, Lumina