

Nuno Alexandre Pinto da Silva

**MULTI-DIMENSIONAL SERVICE-ORIENTED
ONTOLOGY MAPPING**



**Universidade de Trás-os-Montes e Alto Douro
Vila Real, 2004**

Multi-Dimensional Service-Oriented Ontology Mapping

a thesis submitted for the Degree of Doctor of Philosophy by

Nuno Alexandre Pinto da Silva

supervised by

Professor João Manuel Simões Rocha

Professor José Carlos Silva Cardoso

Universidade de Trás-os-Montes e Alto Douro

Vila Real, Portugal

September 2004

ABSTRACT

Ontology mapping is the process whereby semantic relations are defined between two ontologies at the conceptual level, which in turn are applied at data level transforming source ontology instances into target ontology instances. Ontology mapping, as an information integration approach, faces new challenges with the advent of new technological and socio-organizational paradigms such as Semantic Web and Virtual Organizations, especially due to the unprecedented levels of distribution, heterogeneity and evolution.

The first contribution of this thesis is MAFRA – Mapping FRAMework, a systematized interpretation of the ontology mapping process. MAFRA identifies, integrates and organizes the ontology mapping process phases and complemented modules into a meaningful reference model. Due to its wide coverage it further provides a classification artifact for works from distinct but interrelated research fields. While MAFRA identifies several other phases of the process, the rest of the research developed in the scope of this thesis specially focuses on the core phases of the process: the specification and execution of semantic relations.

The Semantic Bridging Ontology (SBO) is the result of the research on analysis, characterization, specification and representation of semantic relations. SBO describes the semantic relations domain of knowledge, providing not only a reasoning mechanism but also a representation and exchange mechanism of semantic relationships, when instantiated. SBO is a very simple and compact ontology, but its extensional structure based on transformation services allows its adoption in very

distinct and complex scenarios. While most of the syntactical relations are provided by transformation services, the structural relations are provided by the entities and their relationships defined by SBO. Furthermore, because SBO clearly and univocally characterizes the different types of semantic relations and their behaviors, it serves as a driving mechanism for the specification of semantic relations. SBO is further complemented by a software application with a graphical user interface that allows the definition and the automatic storage and verification of the semantic relationships.

The semantic relationships resulting from previous phase are then applied in the execution phase. The general-purpose transformation process proposed in this thesis conforms to and exploits the SBO conceptualization, namely concerning the notion of transformation service. The proposed transformation process distinguishes between the generic query and filtering of instances, and the specific transformation provided by the transformation services associated with each semantic relation. The query and filtering phases of the process require specific and extensive processing of data according to the semantic relationships specification, which have been specified recurring to the formal, well-known relational data model algebra, motivating a formal yet very explicit and especially very compact description of the process. Adopting the query-filtering-transformation method, the process exploits and promotes the notion of independent transformation service, allowing the inclusion and modification of the transformation capabilities of the system with no modification of other components of the system.

This independent transformation service approach is extrapolated into the notion of multi-dimensional service in which services capabilities are no longer limited to the transformation of instances but to other phases of the process, namely the semantic relation specification, verification, evolution and negotiation. This new notion of service is further integrated in the multi-dimensional service-oriented architecture, in which services represent and embody specific expertise on the application of the service itself along the semantic relationships life-cycle, providing competencies as requested to the core phases of the process.

The potential advantages of the multi-dimensional service-oriented architecture are tested and exploited in the researched process for automatic definition of semantic relationships. In this case test, services are expanded with competencies to judge the relevance of a set of similarities measures in the definition and confirmation of a semantic relation with which the service is associated.

The proposed ideas and processes have been implemented in the scope of this thesis into the MAFRA Toolkit application, and tested with several ontology mapping scenarios. Additionally, The MAFRA Toolkit has been applied and tested in the scope of several third-party EU-funded projects.

RESUMO ALARGADO

Sendo ontologia uma descrição das características do conteúdo de repositórios de dados, informação ou conhecimento, torna-se possível através da definição de equivalências semânticas entre os elementos das ontologias, relacionar esses repositórios. Mapeamento de ontologias é um processo que consiste na definição a nível ontológico dessas relações semânticas entre entidades de uma ontologia de origem e de uma ontologia de destino. Essas relações são posteriormente aplicadas na transformação das instâncias numa base de conhecimento conforme com a ontologia de origem em instâncias numa base de conhecimento conforme com a ontologia de destino.

O trabalho realizado nesta tese é genericamente dividido em sete partes, que correspondem ao longo da tese a outros tantos capítulos:

1. Motivações;
2. Ontologia;
3. MAFRA – MApping FRAmework;
4. Relacionamento semântico;
5. Execução das relações semânticas;
6. Arquitectura do sistema baseada em serviços multi-dimensionais;
7. Desenvolvimento e experiências.

As secções seguintes descrevem sumariamente o trabalho desenvolvido e os resultados mais relevantes atingidos em cada uma delas, sendo a última secção reservada à apresentação numa

síntese do trabalho desenvolvido e apresentação de alguns indicadores da relevância e utilidade da investigação realizada.

1. Motivações

Mapeamento de ontologias é considerada uma tecnologia fundamental em cenários em que a troca e partilha de informação sejam essenciais. Interoperabilidade entre sistemas de informação, web semântica, organizações virtuais e negócios electrónicos, migração de dados entre sistemas e evolução dos modelos de dados subjacentes aos sistemas de informação, são alguns desses cenários. Através da análise e sistematização das características e das necessidades de interoperabilidade associados a estes cenários, definiram-se os seguintes requisitos dum sistema de mapeamento de ontologias:

1. Identificação, especificação e representação de relações sintácticas, estruturais e semânticas entre ontologias;
2. Transformação da informação transmitida entre intervenientes na comunicação, de acordo com as relações anteriores;
3. Negociação das relações anteriores;
4. Manutenção das relações anteriores;
5. Integração (mas minimização) da participação do ser humano no processo de mapeamento de ontologias, o que sugere a adopção dum sistema semi-automático de mapeamento de ontologias;
6. Adopção de tecnologia e soluções do contexto da web semântica.

Estes requisitos serão posteriormente comparados com a investigação realizada e com os resultados atingidos, o que permite aferir da qualidade da investigação e dos resultados atingidos.

2. Ontologia

Não existe uma definição universalmente aceite de ontologia, mas a definição de Gruber - “ontologia é uma especificação explícita duma conceptualização” – tende a ser genericamente aceite no contexto dos sistemas de informação e conhecimento. No contexto desta tese porém, esta definição é demasiado genérica, o que motiva a análise de características e comparação com outros conceitos comumente relacionados, nomeadamente com o conceito de esquema de base de dados. Da comparação efectuada conclui-se que ambos partilham vários pressupostos e características, mas torna-se contudo evidente que, mesmo não sendo conceptualmente e formalmente universais, as capacidades de descrição e caracterização semântica do domínio de conhecimento de ontologia são superiores às do esquema de base de dados. Partindo desta observação conclui-se que ontologia é, de ambos, o artefacto potencialmente mais capaz de fornecer elementos de raciocínio para a definição de relações semânticas entre repositórios de dados, informação e/ou conhecimento.

Concluindo o tema sobre a caracterização do conceito de ontologias, introduz-se uma especificação formal do modelo de ontologia que será usada no decorrer da tese. Os elementos básicos de ontologia são:

- Conceito, que corresponde à noção de classe na modelação orientada por objectos;
- Propriedade, que corresponde à noção de membro de dados na modelação orientada por objectos. Pode ser especializada em relações (entre conceitos) e atributos (quando o seu valor é primitivo);
- Hierarquia (ou taxonomia) de conceitos, que corresponde à relação hierárquica entre classes na modelação orientada por objectos;
- Axiomas, nomeadamente de caracterização das propriedades (ex. cardinalidade, simetria);
- Léxico, que pela sua associação com conceitos e propriedades do mundo real, aumenta os elementos de raciocínio semânticos da ontologia.

Não sendo uma especificação universal, a formalização apresentada permite uma aceitação suficientemente alargada e com as características suficientes para o desenrolar do trabalho de investigação.

Uma vez definidos o contexto básico e os pressupostos do trabalho a desenvolver, inicia-se a investigação propriamente dita.

3. MAFRA - MApping FRAMework

A primeira contribuição desta tese é o MAFRA – MApping FRAMework. MAFRA é o resultado da análise e sistematização do processo de mapeamento de ontologias, não apenas sobre as suas duas fases principais (a definição e execução das relações semânticas), mas sobre uma perspectiva mais abrangente. Assim, foram identificados dois grupos distintos de tarefas ou componentes do processo:

- Um conjunto de tarefas principais, (implícita ou explicitamente) comuns a qualquer cenário de mapeamento de ontologias, sem as quais o processo não faz sentido:
 - Normalização de léxicos;
 - Avaliação de semelhanças entre ontologias;
 - Definição das relações semânticas;
 - Execução das relações semânticas;
 - Avaliação dos resultados;
- Um conjunto de tarefas complementares ao processo, nomeadamente relacionadas com a automatização das tarefas principais:
 - Evolução das relações semânticas de acordo com as alterações das ontologias e dos

requisitos de aplicação;

- Negociação de relações semânticas novas e de aplicação de outras já existentes;
- Conhecimento e restrições subjacentes a domínios de conhecimento que potenciem os resultados do processo;
- Interface com o utilizador para a manipulação da informação de mapeamento, nomeadamente das relações semânticas.

Uma das ideias fundamentais subjacente ao MAFRA é a de que o conjunto de relações semânticas existente entre duas ontologias (documento de mapeamento) têm um ciclo de vida como um qualquer documento, sistema ou entidade, nomeadamente no que se refere ao processo iterativo de melhoria baseado nos resultados do próprio sistema. Nesse sentido, o comum ciclo de vida de um sistema é adaptado às fases específicas do processo de mapeamento de ontologias, em que:

- As tarefas principais formam um ciclo de melhoria iterativo (e interactivo com o utilizador através da interface);
- As tarefas complementares interagem com todas as tarefas principais sempre que necessário e de acordo com as suas capacidades.

Assim, MAFRA é um modelo de organização do processo de mapeamento de ontologias, mas que devido à sua abrangência e generalidade é também um modelo de referência que permite e facilita a classificação de trabalhos distintos mas de domínios de investigação relacionados.

Embora o MAFRA defina cinco fases principais como compondo o processo de mapeamento de ontologias, o resto do trabalho desenvolvido foca-se nas suas duas fases fundamentais: a definição e a execução de relações semânticas.

4. Relacionamento semântico

A investigação respeitante à definição de relações semânticas focou fundamentalmente dois assuntos:

- A análise, caracterização e sistematização das relações semânticas que potencialmente ocorrem entre ontologias;
- A especificação (conceptual), definição e representação das relações semânticas.

Do primeiro ponto o resultado mais relevante é a sistematização das várias dimensões das relações semânticas:

- Tipos das entidades relacionadas semanticamente, sendo que qualquer combinação que envolva os tipos de entidades previstas no modelo de ontologia são aceites (ex. Conceito-Propriedade, ou Relação-Atributo);

- Transformação, que diz respeito à função necessária para que as instâncias da ontologia de origem sejam transformadas em instâncias da ontologia de destino. Direcionalidade e plenitude são duas sub-dimensões dependentes da função de transformação associada à relação semântica;
- Cardinalidade, que diz respeito ao número de entidades da ontologia de origem e de destino relacionados semanticamente. Os valores desta dimensão variam entre 0:1 e m:n, sendo o primeiro valor relacionado com o número de entidades da ontologia de origem e o segundo com os da ontologia de destino;
- Restrições, que diz respeito às condições que se têm de verificar para que a relação semântica seja executada. As condições são definidas através da combinação de três tipos de elementos: (i) instâncias das entidades relacionadas semanticamente, (ii) instâncias de outras entidades ou (iii) elementos não disponíveis nas ontologias;
- Estrutural, que diz respeito às relações entre relações semânticas, nomeadamente de ordem e composição.

Desta sistematização deduz-se que a única dimensão não universalmente caracterizada é a dimensão de transformação, o que significa que não é possível prever todas as necessidades de transformação. Assim sendo, as capacidades dum sistema de mapeamento de ontologias é dependente da capacidade de especificar, representar e posteriormente executar as transformações necessárias entre entidades ontológicas. É necessário portanto e além disso, que os mecanismos de especificação, representação e execução sejam dotados com capacidades de evolução e adaptação às diferentes necessidades de transformação impostas por diferentes cenários de mapeamento de ontologias.

Entra-se assim na especificação e representação de relações semânticas. A Semantic Bridging Ontology (SBO) é o resultado mais visível da investigação desenvolvida nesta tese sobre este assunto e um dos mais relevantes de toda a tese. A SBO descreve (especifica) o domínio de conhecimento relacionado com as relações semânticas, fornecendo não só um mecanismo de raciocínio mas também, quando instanciada para um determinado cenário, um mecanismo de representação e partilha de relações semânticas. A SBO é uma ontologia simples e compacta, mas a sua estrutura baseada em serviços permite a extensão das capacidades de transformação conforme necessário e de uma forma conceptual, estando assim apta para aplicação em cenários de complexidade distintas, tal como estipulado anteriormente.

Em particular a SBO adopta um modelo:

- Orientado a objectos, pois as relações semânticas entre conceitos das ontologies são modelados numa hierarquia, beneficiando as sub-relações semânticas das propriedades definidas para as super-relações semânticas, através do mecanismo de herança;

- Centrado nas propriedades, pois as relações semânticas entre propriedades são definidas independentemente das dos conceitos. Contudo, como as instâncias das propriedades ocorrem no contexto das instâncias dos conceitos, é necessário que as relações semânticas de propriedades sejam associadas com as de conceitos.

No entanto, foram definidos ainda outros mecanismos:

- Caminhos, que corresponde a um conjunto válido de relações entre conceitos (ex. Pessoa/casado com/Pessoa/nome/String);
- Restrições sobre a relação semântica, usando operadores lógicos (ex. and, or) e de comparação (ex. =, <, >);
- Alternativas mútuas, que permite que, de entre um conjunto de relações semânticas, no máximo uma delas seja executada, em função das restrições de cada uma.

Enquanto alguma da heterogeneidade semântica entre ontologias é ultrapassada através da manipulação sintáctica das instâncias das propriedades, e são portanto suportadas por serviços de transformação específicos, as relações estruturais são suportadas directamente pelos conceitos e interrelações da SBO.

Para além das capacidades de modelação e raciocínio, porque restringe as relações entre entidades explicitamente e univocamente, a SBO serve como mecanismo de orientação no processo de definição de relações semânticas, o que será explorado posteriormente num processo semi-automático de definição de relações semânticas.

5. Execução das relações semânticas

As relações semânticas resultantes da fase anterior são, então, aplicadas na transformação das instâncias.

O resultado mais importante da investigação relacionada com este assunto é o processo genérico de execução de relações semânticas que conformem com a SBO. O processo proposto distingue três fases:

- Fase de interrogação, que diz respeito ao processo de recolha de instâncias da base de conhecimento de acordo com os parâmetros de cada relação semântica. Esta fase corresponde ao maior esforço de investigação deste assunto uma vez que não existe disponível tecnologia de interrogação de bases de conhecimento de ontologias que seja capaz de agregar a informação segundo vários caminhos. Para isso foi desenvolvido um processo baseado na representação de caminhos em árvore. Esta representação permite a combinação dos resultados das interrogações, através da aplicação dos resultados de umas na execução de outras. O resultado desta fase é uma tabela (relacional) em que as colunas representam todos os caminhos definidos

na relação semântica, e os valores são as instâncias acedidas por cada um desses caminhos ao longo do processo de interrogação. O processo desenvolvido foi especificado recorrendo a álgebra relacional, pois trata-se de uma álgebra formal e extensivamente reconhecida. O resultado é uma especificação formal, contudo clara, compacta e de fácil implementação;

- Fase de filtragem, diz respeito ao processo de verificação das condições da relação semântica. O processo ocorre sobre a tabela calculada na fase anterior, e o resultado é uma tabela com as mesmas colunas em que cada linha da tabela respeita as condições impostas. Uma vez que os operadores de comparação são extensíveis, tal como os serviços de transformação, foi também necessário desenvolver especificamente este processo;
- Fase de transformação diz respeito ao processo específico executado pelo serviço de transformação associado a cada relação semântica. O processo de transformação é dependente do serviço, mas os dados a serem processados são produzidos nas fases anteriores independentemente do serviço, o que facilita a sua implementação.

Adoptando uma abordagem por fases (interrogação-filtragem-transformação), o processo explora a noção de serviço de transformação, potenciando a inclusão e modificação das capacidades de transformação do sistema, sem necessidade de alterar outros componentes do sistema. Esta característica será posteriormente explorada aquando da especificação da arquitectura do sistema.

Nesta fase foi ainda desenvolvido um processo que permite ultrapassar determinadas heterogeneidades semânticas impossíveis de ultrapassar com os mecanismos propostos até aqui. Em geral, tais heterogeneidades advêm da adopção de granularidade mais fina por parte da ontologia de destino que a granularidade da ontologia de origem. Por exemplo, quando “endereço” na ontologia de origem é definido como um atributo e na ontologia de destino é definido como um conceito composto por vários atributos (ex. Rua, Código Postal, País). O processo desenvolvido baseia-se na especificação extensional de entidades da ontologia de origem e, tal como o seu nome indica, recorre à classificação das instâncias da base de conhecimento de origem segundo um conjunto de condições baseadas nos seus valores. O processo desenvolvido é baseado no conceito de modelação proposto pelo paradigma Description Logics. A adopção desta abordagem não obriga a usar *skolem terms*, como noutras abordagens, o que por sua vez motivaria a necessidade de ordenação das relações semânticas. Se assim fosse, a complexidade dos processos de definição e de execução das relações semânticas seria necessariamente maior.

Por fim foi desenvolvida uma abordagem de verificação de condições das relações semânticas verificáveis apenas após a fase de transformação, pelo que o processo foi expandido para cinco fases (interrogação-filtragem-transformação-filtragem-instanciação). Como a cardinalidade das entidades da ontologia de destino é uma dimensão da relação semântica paradigmática para este problema, a análise e solução proposta focam esta dimensão em particular. Nesse sentido foram

definidos três novos operadores de comparação de cardinalidade para complementar os anteriormente definidos para a verificação da cardinalidade da base de dados de origem (operadores de comparação), de forma a que não surjam ambiguidades quer na definição quer na execução das relações semânticas.

6. Arquitectura do sistema baseada em serviços multi-dimensionais

A arquitectura do sistema de mapeamento de ontologias proposto nesta tese é baseado na extrapolação da noção de serviço de transformação independente, sugerido na SBO e posteriormente adoptado na fase de execução. A extrapolação ocorre em três dimensões distintas:

- Expansão das competências dos serviços de transformação, para que estes contribuam para outras fases do processo, na aplicação do serviço a relações semânticas. Porque os serviços são agora dotados de competências em diversas fases do processo, são referidos por serviços multi-dimensionais;
- Tornar os serviços entidades fornecedoras de competências (*know-how*), independentes de qualquer módulo ou fase do processo. A relação entre as fases do processo e os serviços evolui portanto para uma perspectiva de cooperação em que os serviços participam (de acordo com as suas competências) na melhoria da sua própria aplicação nas relações semânticas;
- Tornar os serviços entidades dinâmicas e auto-descriptivas, capazes de promover a adaptação do sistema a diferentes situações de mapeamento. Os serviços tornam-se assim entidades externas ao sistema, mas com capacidade de se associarem (*pluggable*) ao sistema automaticamente e sem necessidade de alteração de outros componentes do sistema.

A arquitectura do sistema desenvolvida nesta fase, denominada Multi-dimensional Service-oriented Architecture, adopta e promove este novo conceito de serviço como entidade fundamental no processo de mapeamento. Nesta arquitectura os serviços adquirem, representam e fornecem competências até agora representadas e fornecidas por peritos/utilizadores. Devido à modularidade dos serviços, o *know-how* dos peritos é modelado em múltiplos módulos, evoluindo e adaptando-se às necessidades de novos cenários de mapeamento independentemente de outros serviços e de outros componentes do sistema, o que parece ser benéfico na resposta às características de distribuição e dinamismo intrínsecas à web semântica. Evolução, negociação e relacionamento semântico (semi-)automático são apontados como fases do processo de mapeamento que mais vantagens potencialmente tiram da arquitectura proposta. Com o intuito de analisar e testar os benefícios e potencialidades da arquitectura proposta, decidiu-se aplicá-la na semi-automatização do processo de relacionamento semântico.

A investigação realizada no âmbito deste caso de teste está longe de se limitar à aplicação da arquitectura, tendo a investigação obtido resultados importantes na forma como o problema da automatização da definição das relações semânticas é analisado e abordado.

De uma forma genérica o processo desenvolvido usa entidades independentes (*matchers*) para a avaliação de semelhanças entre entidades das duas ontologias (*matches*). Dependendo da decisão do perito/utilizador (ou outra entidade responsável e competente nessa matéria), um conjunto de *matches* poderá dar origem a uma relação semântica. Cada *match* caracteriza a semelhança entre duas entidades das ontologias de acordo com uma determinada dimensão (ex. semelhança de nomes) o que é claramente insuficiente para atingir resultados aceitáveis. Assim, em vez de utilizar apenas um tipo de *match* como normalmente acontece noutras abordagens, o processo desenvolvido durante esta tese sugere a adopção de múltiplos tipos de *matches* (e portanto de *matchers*) e a combinação das suas avaliações numa única, de acordo com as condições específicas definidas para/por cada serviço. Assim, para além de se poder utilizar novas formas de avaliação de semelhanças (novos *matchers*), é possível fazer depender do serviço a decisão de ser associado a determinada relação semântica.

Em mais detalhe o processo desenvolvido é composto por três fases:

- Definição de relações semânticas provisórias (*clusters*) através da agregação de *matches*, de forma que os tipos e cardinalidade das entidades dos *matches* respeitem a interface dos serviços disponíveis no sistema. A cada *cluster* é associado o serviço cuja interface permite/sugere o *cluster*;
- Confirmação (ou anulação) de cada um dos *clusters* através das condições que os *matches* devem verificar, e que são especificamente definidas pelo serviço associado ao *cluster*;
- Transformação dos *clusters* em relações semânticas e suas interrelações válidas e executáveis. De referir que nesta fase é plenamente adoptado o modelo definido na SBO, pelo que todas as interrelações entre relações semânticas previsto na SBO, mesmo as não fundamentais (ex. em modelos orientados a objectos a adopção do conceito de hierarquia entre classes é aconselhável mas não obrigatório), são definidas e os seus benefícios explorados. Por isso, são por vezes inferidas novas relações semânticas, e que dão origem a novos *clusters* e *matches* respectivos, para manter o processo consistente.

As condições auto-definidas pelo serviço podem evoluir ao longo do tempo e em função de muitos factores, incluindo processos de aprendizagem por observação do comportamento do perito.

7. Desenvolvimento e experiências

Se bem que a parte teórica, descrita até agora, tenha sido a que requereu mais esforço e dedicação temporal, a parte de implementação e experimentação foi também muito importante, tanto mais que decorreu em paralelo com a parte teórica durante a maior parte do tempo de implementação.

Esta relação tão próxima entre implementação e investigação teórica é considerada neste trabalho benéfica para as duas componentes, em particular porque permitiu:

- Provar a viabilidade e utilidade dos resultados da investigação teórica;
- Fornecer opiniões sobre as competências e limitações das propostas da investigação teórica;
- Promover e divulgar as ideias a uma comunidade mais alargada;
- Fornecer uma ferramenta funcional que pudesse ser usada em projectos de terceiros, por forma a receber mais e fundamentadas opiniões sobre as ideias propostas, baseadas em experiências sobre cenários diferentes.

Foi portanto desenvolvida uma aplicação informática denominada MAFRA Toolkit, e que constitui o mais relevante resultado desta fase. Se bem que a grande maioria das funcionalidades implementadas no MAFRA Toolkit sejam o resultado da investigação teórica descrita, determinadas funcionalidades foram implementadas segundo uma abordagem pragmática. Esta observação é especialmente verdadeira no que se refere ao desenvolvimento da interface gráfica. Esta, tal como o resto, é fortemente influenciada pela adopção do KAON Workbench como tecnologia para a manipulação de ontologias e bases de conhecimento. Apesar de muitas competências relevantes o KAON Workbench não tem suporte para linguagens de interrogação, mas, porque mesmo as linguagens de interrogação não solucionariam o problema de interrogação encontrado, as capacidades de manipulação disponíveis acabaram por ser suficientes. Contudo, estas limitações acabaram por conduzir a um processo de implementação bastante moroso.

No entanto o MAFRA Toolkit e as ideias preconizadas nesta tese estão agora estáveis e funcionais, tendo sido extensivamente e com sucesso aplicados em projectos terceiros, o que permitiu inferir com algumas evidências sobre a viabilidade e relevância da investigação realizada.

8. Resultados atingidos

Embora conclusões formais não possam ser retiradas devido principalmente à incapacidade de formular o problema completamente (a dimensão de transformação não pode ser completamente definida), é possível, através da comparação dos requisitos estipulados no início da tese bem como de indicadores usados pela comunidade científica, inferir que a qualidade da investigação e dos resultados atingidos é satisfatória.

Em particular para os requisitos enunciados no final da análise das motivações, a investigação realizada forneceu o seguinte suporte:

1. Identificação, especificação e representação de relações sintácticas, estruturais e semânticas entre ontologias. Este requisito é extensivamente suportado pela investigação realizada:
 - O MAFRA, em que identifica explicitamente a fase de relacionamento semântico;

- A SBO, que especifica e permite a definição e representação de relações sintácticas, estruturais e semânticas. Embora seja conceptualmente impossível determinar o grau de suporte fornecido, as experiências demonstram a sua competência num grande número de aplicações e cenários;
 - A identificação das relações é suportada pelo processo semi-automático de relacionamento semântico desenvolvido como caso de teste de aplicação da arquitectura baseada em serviços multi-dimensionais. Embora o processo tenha por objectivo sugerir um conjunto razoável e válido de relações semânticas, é perceptível que a solução carece ainda de maior refinamento das condições definidas pelos serviços e porventura do desenvolvimento e adopção de novos *matchers*;
2. Transformação da informação transmitida entre intervenientes na comunicação, de acordo com as relações anteriores. Este requisito é referido em duas fases da investigação:
 - Na sistematização do processo de mapeamento de ontologias no MAFRA, em particular na fase de execução das relações semânticas;
 - Na investigação realizada sobre o processo de execução;
 3. Negociação das relações anteriores é suportada pelo módulo de Negociação do MAFRA. Apesar de esta ser referida como uma tarefa que poderá beneficiar com a adopção da arquitectura baseada em serviços multi-dimensionais, nenhuma investigação sistemática foi realizada sobre este tópico;
 4. Manutenção das relações anteriores é suportada pela tarefa complementar do MAFRA denominada Evolução, sendo que, tal como para a negociação, nenhuma investigação sistemática foi realizada sobre este assunto;
 5. Integração (mas minimização) da participação do ser humano no processo de mapeamento de ontologias, o que sugere a adopção dum sistema semi-automático de mapeamento de ontologias. Este requisito é parcialmente suportado através dos seguintes elementos:
 - Pelo módulo de Conhecimento e restrições subjacentes a domínios de conhecimento do MAFRA, que representa conceptualmente todas as fontes de conhecimento e *know-how* que possam ser úteis para a automação do processo de mapeamento de ontologias;
 - Os serviços multi-dimensionais e os *matchers* são algumas destas fontes de conhecimento explicitamente usadas nas soluções propostas;
 - O processo semi-automático de relacionamento semântico é contudo o mais relevante suporte deste requisito, pois combina as fontes de informação (serviços e *matchers*) através dum método e regras (heurísticas) de relacionamento semântico de entidades. Se o processo reduz a participação do ser humano no processo, por outro lado permite e sugere o aperfeiçoamento por parte do utilizador das relações semânticas definidas automaticamente;

- A forma declarativa, simples e compacta da SBO tem por objectivo reduzir os esforços do ser humano em definir e representar as relações semânticas;
 - A interface gráfica do MAFRA Toolkit fornece um mecanismo de interacção entre o suporte automático e o ser humano, permitindo-lhe ao mesmo tempo a sua participação e redução do esforço de participação.
6. Adopção de tecnologia e soluções do contexto da web semântica. Este requisito é amplamente referido e considerado durante a tese, nomeadamente:
- A SBO é representada em RDFS e DAML+OIL, duas das mais importantes linguagens de representação de ontologias no contexto da web semântica;
 - O conjunto de relações semânticas definidas entre duas ontologias é representado através de RDF, o modelo de representação de base de todas as linguagens de representação na web semântica;
 - A arquitectura baseada em serviços multi-dimensionais, nomeadamente a noção de serviços independentes, dinâmicos e auto-descritivos é adequada às características da web semântica.

É portanto perceptível que enquanto o nível de suporte fornecido é difícil de determinar, por outro lado todos os requisitos foram alvo de investigação e suporte, e a grande maioria é, pelo menos parcialmente, suportado pela aplicação desenvolvida.

Adicionalmente, outros indicadores científicos servem para confirmar a opinião defendida de que um trabalho válido e útil foi desenvolvido:

- O grande número de conferências e jornais internacionais com revisão para as quais o trabalho desenvolvido nesta tese foi aceite para publicação;
- O grande número de publicações científicas que citam o MAFRA, SBO, o processo de execução e o MAFRA Toolkit em diferentes campos de investigação;
- As boas opiniões recebidas e do grande número de aplicações e experiências realizadas com a SBO e o MAFRA Toolkit por/em projectos terceiros, incluindo projectos comerciais;
- As sugestões construtivas recebidas no sentido de continuar a investigação e a implementação do MAFRA Toolkit.

Assim, embora seja difícil concluir formalmente acerca da validade das ideias propostas, é possível concluir que os resultados atingidos são úteis e relevantes para a comunidade científica, e a breve prazo para soluções comerciais.

TABLE OF CONTENTS

Abstract	iii
Resumo Alargado	v
1. Motivações	vi
2. Ontologia	vi
3. MAFRA - MApping FRAmework	vii
4. Relacionamento semântico	viii
5. Execução das relações semânticas	x
6. Arquitectura do sistema baseada em serviços multi-dimensionais	xii
7. Desenvolvimento e experiências	xiii
8. Resultados atingidos	xiv
Table Of Contents	xvii
Table Of Figures	xxv
Table Of Examples	xxvii

First Part	1
Chapter 1 Introduction	3
1.1 Context	3
1.2 Thesis organization	5
Chapter 2 Motivations	7
2.1 Schema integration	8
2.1.1 Database integration	9
2.1.2 Data warehousing	12
2.1.3 Data warehousing Vs. Database integration	15
2.2 Semantic Web	15
2.3 Virtual Organizations and E-Business	18
2.3.1 Agent-based Systems	19
2.3.2 Web services	20
2.3.3 Virtual organization and E-Business requirements	21
2.4 Knowledge management	22
2.5 Outlook	23
2.5.1 Mapping specification phase	24
2.5.2 Mapping transformation phase	24
2.5.3 Human intervention	25
2.5.4 Common consensus building	25
2.5.5 Evolution	25
2.5.6 Semantic Web importance	25
2.5.7 Summary of requirements	26
Chapter 3 Ontology	27
3.1 Characterization of ontologies	28
3.1.1 Generality	28
3.1.2 Granularity	29
3.1.3 Formality	30
3.1.4 Roles	31
3.1.5 Miscellaneous characteristics	32
3.2 Ontology Vs. Database schema	32
3.2.1 Overview of informal characteristics	34

3.3 Formal Definition of Ontology	35
3.3.1 Schematic layer	36
3.3.2 Lexical layer	37
3.3.3 Axiomatic layer	38
3.3.4 Formal definition of Knowledge Base	39
3.3.5 Example 3.4 - Simple ontology and knowledge base	40
3.4 Summary	41
Second Part	43
Chapter 4 Ontology Mapping Framework	45
4.1 Quality vectors	46
4.2 MAFRA Overview	46
4.3 Horizontal Dimension of MAFRA	49
4.3.1 Lift & Normalization	49
4.3.1.1 Lift	49
4.3.1.2 Normalization	50
4.3.2 Similarity Measuring	52
4.3.3 Semantic Bridging	56
4.3.3.1 Automation	56
4.3.3.2 Specification methods	57
4.3.3.3 Representation language	58
4.3.3.4 Outlook of Semantic Bridging	59
4.3.4 Execution	59
4.3.4.1 Classification process	59
4.3.4.2 Transformation process	61
4.3.4.3 Entity-driving execution	61
4.3.4.4 Operation mode	62
4.3.4.5 Outlook of Execution	63
4.3.5 Post-processing	63
4.4 Vertical Dimension of MAFRA	65
4.4.1 Evolution	65
4.4.2 Cooperative Consensus Building	65
4.4.3 Domain Constraints and Background Knowledge	66
4.4.4 Graphical User Interface	67
4.5 Ontology mapping process flow	68

4.6 Summary	68
Chapter 5 Semantic Bridging	69
5.1 Ontology mapping: two-phases process	70
5.1.1 Informal definition	70
5.1.2 Formal definition	71
5.2 Semantic heterogeneity	71
5.2.1 Entity type dimension	73
5.2.2 Transformation dimension	74
5.2.2.1 Function	74
5.2.2.2 Directionality	75
5.2.2.3 Completeness	76
5.2.3 Cardinality dimension	76
5.2.4 Constraint dimension	77
5.2.5 Structural dimension	78
5.2.6 Summary of characterization	78
5.3 State of the art	80
5.3.1 Protégé	81
5.3.2 Stuckenschmidt and colleagues	82
5.3.3 RDFT	85
5.3.4 OntoMerge	86
5.3.5 Summary	88
5.4 Semantic Bridging Ontology	91
5.4.1 SBO Overview	92
5.4.2 Service	93
5.4.3 Semantic Bridge	94
5.4.3.1 Concept Bridge	96
5.4.3.2 Property Bridge	96
5.4.4 Path	97
5.4.4.1 Step	98
5.4.4.2 Path	98
5.4.4.3 Directionality	100
5.4.4.4 Alternative notation	101
5.4.4.5 Outlook of Path	101
5.4.5 Condition Expression	101
5.4.6 Array	103

5.4.7 Ontology Mapping Document	104
5.4.7.1 Alternative Bridges of ConceptBridges	105
5.4.7.2 Alternative Bridges of PropertyBridges	106
5.4.7.3 Relation between ConceptBridges and PropertyBridges	106
5.4.7.4 Hierarchy of ConceptBridges	108
5.5 Example 5.21 – Semantic bridging annotated example	110
5.5.1 Ontology Mapping Document	111
5.5.2 ConceptBridges	111
5.5.3 ConditionExpressions	111
5.5.4 Disjoint bridges	112
5.5.5 Property Bridges	112
5.5.6 Object-Oriented modeling	114
5.6 Conclusions	115
Chapter 6 Execution process	117
6.1 Execution process overview	118
6.1.1 ConceptBridge execution	119
6.1.2 PropertyBridge execution	121
6.2 Internal process	124
6.2.1 Querying the source knowledge base	124
6.2.1.1 Tree-based representation of Paths	131
6.2.1.2 Tree-based query	133
6.2.2 Filter the knowledge base query	136
6.2.3 Create the target knowledge base instances	136
6.2.4 Example 6.17 – Execution process annotated example	138
6.2.4.1 ConceptBridge	139
6.2.4.2 PropertyBridge	140
6.2.4.3 Inter-relation of instances	144
6.3 Extensional Specification	146
6.3.1 Example 6.18 - ConceptBridges with 1:n cardinality	147
6.3.2 Analysis of the problem	149
6.3.3 Developed approach	152
6.3.3.1 ConceptBridge and PropertyBridge	154
6.3.3.2 Semantics of the Extensional Specification arguments	155
6.3.3.3 Extensional Specification and Description logics	158

6.3.4 Example 6.24 - Extensional specification annotated example	158
6.4 Constraints upon target instances	163
6.4.1 Analysis of the problem	163
6.4.2 Developed solution	165
6.4.3 Example 6.27 - Cardinality annotated example	167
6.5 Conclusions	169
Chapter 7 Multi-Dimensional Service-Oriented Architecture	173
7.1 Observations	174
7.1.1 The gap between similarity measuring and semantic bridging	174
7.1.2 Cooperative consensus building	175
7.1.3 Evolution	176
7.1.4 Synthesis	177
7.2 Proposal	177
7.3 Automatic Semantic Bridging	179
7.3.1 Observations	179
7.3.2 Hypothesis	179
7.3.3 Specification	180
7.3.3.1 Matchers and Matches	180
7.3.3.2 Cluster and Clustering	182
7.3.4 Reducing combinatorial space by Service-based clustering	184
7.3.4.1 Example 7.10 – Service-based clustering annotated example	187
7.3.4.1.1 CopyInstance	189
7.3.4.1.2 CopyRelation	189
7.3.4.1.3 CopyAttribute	189
7.3.4.1.4 CountProperties	190
7.3.4.1.5 Split	190
7.3.4.1.6 Concatenation	191
7.3.4.1.7 Example overview	191
7.3.4.2 Service interface conforming vs. non-conforming matches	191
7.3.4.3 CopyInstance specificity	192
7.3.4.4 Outlook of the method	193
7.3.5 Improving Services judgment capabilities	194
7.3.5.1 Proposed approach	194
7.3.5.2 Outlook of the proposed approach	196

7.3.6 Automatic definition of the ontology mapping document	197
7.3.6.1 Definition of ConceptBridges	197
7.3.6.2 Definition of \prec -relationships	198
7.3.6.3 Definition of PropertyBridges	198
7.3.6.4 Definition of \diamond -relationships	200
7.3.6.5 Definition of AlternativeBridges	202
7.3.6.6 Outlook of the automatic definition of the ontology mapping document	202
7.3.7 Outlook of the automatic bridging process	203
7.4 Summary	204
Chapter 8 Development and Experiences	207
8.1 Development	207
8.1.1 Ontology and Knowledge Base manipulation	208
8.1.2 Semantic bridging	210
8.1.3 Execution engine	211
8.1.3.1 Tree-based query	212
8.1.3.2 Filtering	212
8.1.3.3 Transformation engine	214
8.1.3.4 Services	217
8.1.4 Graphical User Interface	219
8.1.4.1 Tree-based user interface	219
8.1.4.2 Net-based user interface	222
8.1.5 Outlook	226
8.2 Application experiences	226
8.2.1 Harmonise and Harmo-TEN	227
8.2.2 Artemis and Satine	229
8.2.3 BRIDGE-IT	230
8.2.4 Outlook	231
8.3 Performance and comparison experiences	231
8.4 Conclusion	234
Third Part	235
Chapter 9 Conclusion	237
9.1 Outlook of the thesis	238
9.1.1 Contextualization and motivations	238

Table Of Contents

9.1.2 Theoretical research _____	239
9.1.3 Development and experiences _____	241
9.2 Summary of research achievements _____	242
9.3 Final remarks _____	245
Chapter 10 Ongoing and Future Research _____	247
10.1 Combination of Services _____	248
10.2 Abstraction of the extensional specification elements _____	249
10.3 Automatic semantic bridging process _____	251
10.4 Graphical user interface _____	251
10.5 Evolution _____	252
10.6 Common Consensus Building _____	253
10.7 Integration with/in other systems _____	253
10.8 Development and code re-engineering _____	254
10.9 Standardization _____	255
10.10 Experiences and case tests _____	255
10.11 Outlook _____	256
Fourth Part _____	257
Annex 1 Relational Data Model _____	259
A 1.1 Building blocks _____	260
A 1.2 Relational data model Vs. Ontology data model _____	262
A 1.3 Translating ontology and knowledge base into relational model _____	263
A 1.3.1 Translation of ontology entities into relation schema _____	263
A 1.3.2 Normalizing concept instances _____	263
A 1.3.3 Representation of forward and backward Paths in the relation schema _____	264
A 1.4 Relational algebra _____	264
A 1.5 Complementary operations _____	268
A 1.6 Outlook _____	270
Bibliography _____	271

TABLE OF FIGURES

Figure 2.1 - Five-level multi-database integration architecture _____	10
Figure 2.2 – Data warehousing five-level architecture _____	13
Figure 3.1 – UML representation of the previous ontology and knowledge base _____	41
Figure 4.1 – MAFRA – MApping FRAmework _____	48
Figure 4.2 - Figurative representation of the input and output of the Lift sub-process _____	50
Figure 4.3 – Graph-based similarity measuring scenario _____	53
Figure 4.4 – Similar scenario but providing meaningful entities labels _____	53
Figure 4.5 – Taxonomy of schema matching approaches _____	55
Figure 5.1 – Informal representation of ontology mapping _____	70
Figure 5.2 - UML representation of two ontologies _____	84
Figure 5.3 - UML representation of SemanticBridge and Service conceptual relation _____	92
Figure 5.4 – UML representation of the SemanticBridge and Service conceptual relation _____	93
Figure 5.5 – UML representation of the SBO taxonomy of semantic bridges _____	95
Figure 5.6 – Path between two structurally different ontologies _____	99
Figure 5.7 – Ontology mapping scenario dealing with inverse properties _____	100
Figure 5.8 – UML representation of the SBO relations between SemanticBridges _____	105
Figure 5.9 – Hierarchical relations between ConceptBridges _____	109
Figure 5.10 - Excerpt of Gedcom and Gentology ontologies, represented in UML notation _____	110

Table Of Figures

Figure 6.1 – Simple representation of the execution process _____	118
Figure 6.2 – UML-like representation of ontology mapping scenario _____	120
Figure 6.3 – PropertyBridges representation in UML _____	122
Figure 6.4 – Schematic representation of the left-join attributes in a forward Step _____	126
Figure 6.5 – Schematic representation of left-join attributes in a backward Step _____	127
Figure 6.6 – UML representation of ontology _____	131
Figure 6.7 – Querying KB through Paths with and without common sub-Paths _____	135
Figure 6.8 – ConceptBridges between excerpts of TourinFrance and SIGRT ontologies _____	139
Figure 6.9 – Several SemanticBridges between TourinFrance and SIGRT ontologies _____	141
Figure 6.10 – Some instances of the TourinFrance ontology _____	142
Figure 6.11 – Correlating source and target concept instances through FQ and TI^2 _____	146
Figure 6.12 – Excerpts of SemanticBridges between SIGRT and TourinFrance ontologies _____	147
Figure 6.13 – Ambiguous correlation process _____	149
Figure 6.14 – 1:n Property to Concept semantic relations _____	150
Figure 6.15 – Excerpt of the SIGRT-TIF mapping scenario _____	158
Figure 6.16 – Univocal correlation through extensional specification _____	162
Figure 7.1 – Multi-Dimensional Service-Oriented Architecture _____	178
Figure 7.2 – Simple ontology mapping scenario using UML notation _____	181
Figure 7.3 – Service-based clustering for reduce combinatorial space _____	185
Figure 7.4 – Abstract scenario representing an inferred match _____	199
Figure 7.5 – Automatic semantic bridging without exploiting the properties inheritance _____	201
Figure 7.6 – Automatic semantic bridging when exploiting the properties inheritance _____	202
Figure 7.7 – MAFRA emphasis on Domain Knowledge & Constraints module relations _____	204
Figure 8.1 – Semantic Web technological layers according to Berners-Lee _____	208
Figure 8.2 – UML representation of core classes and interfaces of MAFRA Toolkit _____	218
Figure 8.3 – Screenshot of the KAON SOEP tree-based interface _____	220
Figure 8.4 – MAFRA Toolkit UI: first tree-based implemented interface _____	220
Figure 8.5 – MAFRA Toolkit UI: distinct panels for SemanticBridges and their parameters _____	221
Figure 8.6 – MAFRA Toolkit UI: net-based representation of entities _____	222
Figure 8.7 – MAFRA Toolkit UI: simultaneous representation of all types of entities _____	224
Figure 10.1 – Execution order between SemanticBridges _____	248

TABLE OF EXAMPLES

Example 2.1 – Web technology without semantic awareness _____	16
Example 2.2 – KQML performatives: same content but different meaning _____	22
Example 3.1 – Hierarchical relation in an ontology _____	36
Example 3.2 – Ontology axioms in the form of modeling constructs _____	38
Example 3.3 - Concept inference based on axioms _____	39
Example 3.4 - Simple ontology and knowledge base _____	40
Example 4.1 – Normalization without semantic commitment _____	50
Example 4.2 – Normalization with semantic commitment _____	51
Example 4.3 – Graph-based similarity measuring scenario _____	52
Example 4.4 – Meaning senses of “person” according to Merriam-Webster dictionary _____	54
Example 4.5 – Inconsistencies between semantic bridges and the target ontology _____	64
Example 5.1 – Several transformation functions _____	74
Example 5.2 –Inverse functions: concatenation vs. split _____	75
Example 5.3 – Transformation completeness: loss of information _____	76
Example 5.4 – Stuckenschmidt and colleagues semantic relations _____	84
Example 5.5 – OntoMerge Concept to Concept semantic relations _____	86
Example 5.6 – OntoMerge conditional Concept to Concept semantic relations _____	87
Example 5.7 – OntoMerge Property to Property semantic relations _____	88
Example 5.8 – Definition (instantiation) of a Service _____	94

Table Of Examples

Example 5.9 – Multiple ranges of properties _____	97
Example 5.10 – Path is required between two structurally different ontologies _____	98
Example 5.11 – Path representation _____	99
Example 5.12 – Ontology mapping scenario requiring inverse Path _____	100
Example 5.13 – Definition of a backward Step _____	101
Example 5.14 – Definition of a backward Path _____	101
Example 5.15 – Definition of Paths using the simplified notation _____	101
Example 5.16 – Definition of a ConditionExpression _____	102
Example 5.17 – Generalization of Services parameters _____	103
Example 5.18 – Definition of Service using Arrays _____	103
Example 5.19 – AlternativeBridges of ConceptBridges _____	106
Example 5.20 – Hierarchy of ConceptBridges _____	109
Example 5.21 – Semantic bridging annotated example _____	110
Example 5.22 – Characterization of Service according to its inverse Service _____	116
Example 6.1 – Execution process of ConceptBridges _____	120
Example 6.2 – Execution process of PropertyBridges _____	122
Example 6.3 – Querying a single-Step Path _____	125
Example 6.4 – Left-join operation of a forward Step _____	126
Example 6.5 – Left-joint operation of a backward Step _____	127
Example 6.6 – Querying through a multi-Step backward Path _____	129
Example 6.7 – Querying multiple Paths without addressing referential constraint _____	129
Example 6.8 – Paths with common sub-Paths _____	131
Example 6.9 – Tree-based representation of Paths _____	132
Example 6.10 – Array-like access to tree-based represented Paths _____	132
Example 6.11 – Multi-dimension array-like access to tree-based represented Paths _____	133
Example 6.12 – Retrieving distinct Branches of tree-based represented Paths _____	133
Example 6.13 - Retrieving the common Path of a tree-based represented Path _____	133
Example 6.14 – Querying KB through multiple tree-based represented Paths _____	135
Example 6.15 – Useless query generated columns _____	135
Example 6.16 – Filtering queries _____	136
Example 6.17 – Execution process annotated example _____	138
Example 6.18 - ConceptBridges with 1:n cardinality _____	147
Example 6.19 – Property to Concept semantic relations _____	149
Example 6.20 – Constrained Property to Concept semantic relations _____	150
Example 6.21 – Fine vs. Coarse grained ontology _____	151
Example 6.22 – Different perspective of the same concept instance _____	153
Example 6.23 – Structural interpretation of ConditionExpressions _____	157
Example 6.24 - Extensional specification annotated example _____	158

Example 6.25 – Service-dependent cardinality _____	163
Example 6.26 – Or-based ambiguous cardinality constraints _____	166
Example 6.27 - Cardinality annotated example _____	167
Example 7.1 - Schematic evolution of ontologies demand SemanticBridges evolution _____	176
Example 7.2 - Semantic evolution of ontologies demand SemanticBridges evolution _____	176
Example 7.3 – Matchers and the different dimensions of ontologies _____	180
Example 7.4 – Similarity measuring scenario _____	181
Example 7.5 – Matches forming a cluster _____	182
Example 7.6 – MOMIS generated cluster _____	183
Example 7.7 – Service-Cluster association _____	184
Example 7.8 – Clustering constrained by the Service cardinality _____	184
Example 7.9 – Clustering constrained by the type of arguments of Service _____	184
Example 7.10 – Service-based clustering annotated example _____	187
Example 7.11 – Improving Services judgment capabilities _____	195
Example 7.12 – Confirming a cluster according to the Service requirements _____	195
Example 7.13 – Dismissing a cluster according to the Service requirements _____	196
Example 7.14 – Refining Service requirements _____	196
Example 7.15 – Inferring matches between concepts based on PropertyBridges _____	199
Example 7.16 – Definition of \diamond -relationships _____	200
Example 7.17 – Services commonly used alternatively _____	202
Example 8.1 – Verification process when \diamond -relating two SemanticBridges _____	211
Example 8.2 – RDF definitions of the Equal and Less Operators _____	213
Example 8.3 – Ambiguous appearance of graphical buttons _____	225
Example A 1.1 - Generic transformation of a knowledge base into relations _____	260
Example A 1.2 – Normalization of concept instances _____	263
Example A 1.3 – Table-based representation of concept instances _____	264
Example A 1.4 – Selection operation _____	265
Example A 1.5 – Projection operation _____	265
Example A 1.6 – Cartesian Product operation _____	266
Example A 1.7 – Union operation _____	266
Example A 1.8 – Difference operation _____	267
Example A 1.10 – Intersection operation _____	268
Example A 1.11 – Theta Join operation _____	268
Example A 1.12 – Natural Join operation _____	269
Example A 1.13 – Left Join operation _____	269

FIRST PART

Chapter 1

INTRODUCTION

This chapter describes the technological and socio-organizational contexts in which the work described in this thesis has started.

1.1 Context

Globalization, social and environmental pressures and technological complexity are some of the most important evolutionary challenges that socio-organizational systems (e.g. banking, commerce, manufacturing, education, recreation) and their technological supporting systems, especially the information and communication systems, are currently facing.

Systems are recommended to combine flexibility and adaptability with agility, information with knowledge, autonomy with cooperation, reaction with partnership [Silva, 1998]. Knowledge-based interoperability assumes a central role in the evolution of the systems as it promotes internal agility and supports faster, better and cheaper implementation of partnerships [Neches *et al.*, 1991]. Accordingly, information-based organizations should evolve to encompass semantics and

pragmatics into its information models, converting them selves into knowledge-based organizations.

The first fact motivating the work described in this thesis occurred by the end of 1998 when developing a specific agent-based system for scheduling assistance to manufacturing. It has been noticed the incapacity of that system to deal with agents arriving from different information communities. In fact, the exchange of messages occurring in the system assumed that the content of messages was always conforming to the specific information model of the information community. Despite agents prevented misunderstandings in the system by rejecting unknown information contents, the described situation emphasized the need to overcome information heterogeneity in an automatic and reliable form. Considering the envisaged features of knowledge in promoting understanding between autonomous entities, the application and exploitation of knowledge-based exchange of messages has been suggested thereafter.

In the early stages of this thesis, the main goal has been set to develop and apply knowledge-oriented reconciliation mechanisms to semi-automatically unify (also referred as merging [Pinto *et al.*, 1999]) the content of the agent-based system messages [Silva & Rocha, 2002]. While the development of reconciliation mechanisms remained a valid goal, three important facts motivated the reassessment of research strategy:

- Unification is not always possible. In fact, unification is possible only in very special circumstances, namely when the knowledge of an entity completely overlaps the knowledge of the other entity, and vice-versa. In other circumstances, unification leads to poor integration;
- The technological maturity of reconciliation mechanisms was insufficient to satisfactorily apply it semi-automatically to very demanding scenarios as manufacturing agent-based systems;
- Reconciliation mechanisms are orthogonal technology for many different research fields and applications. Database integration, E-business and Semantic Web for example, are simultaneously relevant application domains and important source of valuable research in the field.

As consequence, the research evolved from the agent-based systems interoperability scenario to a more generic scope, in which the research and development of reconciliation mechanisms capable to cope with information heterogeneity in a larger set of scenarios became the main goal.

Ontology, as an artifact to represent and share characterization of information, is envisaged [Benjamins *et al.*, 1998; Decker *et al.*, 1999; Fensel, 2001; Fensel *et al.*, 2003; Gruber, 1993a; Klusch, 2001; Studer *et al.*, 1998; Uschold *et al.*, 1998] as the foundation element for this endeavor. Ontology combines a set of important characteristics that makes it particularly supportive to different knowledge manipulation tasks such as acquisition, reuse, maintenance, reasoning and exchange.

Ontologies, indeed, arose in recent years as a core mechanism in many different computer science domains, especially in those where knowledge manipulation and exchange are fundamental requirements (further details on Ontology can be found in Chapter 3).

The basic idea behind this work suggests that any entity commits its messages content to one or more non-contradictory ontologies that are shared and interrelated by the interoperability partners or by an especially dedicated entity. The output of the interrelation process is a set of equivalence relations between the ontologies of the entities. These equivalences relations are then applied in transforming the content of the messages from one entity into the content of the messages understood by another entity. The process of identification, specification and application of equivalence relations between two ontologies is referred to as Ontology Mapping.

The goal of this thesis is therefore especially focused on researching a semi-automatic ontology mapping process that would facilitate the knowledge base interoperability between heterogeneous entities. A set of tools implementing the researched ideas will be developed in order to evaluate their feasibility and usefulness, especially in the scope of the Semantic Web.

1.2 Thesis organization

This thesis is composed by ten chapters, one annex and the bibliography, grouped into four general parts:

1. The first part includes Chapter 1 through Chapter 3:
 - Chapter 1 describes the context of research and first motivations.
 - Chapter 2 extensively presents motivations scenarios in which the ontology mapping technology is advisable and beneficial. According to the analysis of motivating scenarios, a systematization of requirements is drawn, serving as reference for the rest of the research.
 - Chapter 3 firstly analyzes and characterizes the concept of ontology. Later, a comparison between ontology and the concept of database schema is presented. Due to the ambiguous understanding of ontology, a formal model of ontology and respective knowledge base are presented, which are adopted as the definitive specifications in the scope of this thesis.
2. The second part is concerned with the description of the research work developed during this thesis, and includes Chapter 4 through Chapter 8:
 - Chapter 4 describes the MAFRA - MApping FRAMework, a systematic interpretation of the overall ontology mapping process, considered the foundational starting point for the rest of the research and development work in this thesis, and one of the novelties proposed in this thesis.
 - Chapter 5 concerns with the characterization, systematization, specification, definition and representation of the semantic relations. SBO – Semantic Bridging Ontology is the major

outcome of this research subject and one of the most important outcomes of this thesis. SBO is not only the specification of the conceptualization of the semantic bridging domain of knowledge, but also the representation and exchange mechanism of semantic relations.

- Chapter 6 presents the execution process researched in the scope of this thesis, which conforms to the SBO. The general-purpose execution process is formally specified based on several primitive operations from the relational data model, resulting in a very explicit and very compact description of the process.
 - Chapter 7 proposes and presents a specific architecture of the ontology mapping system, named Multi-dimensional Service-oriented Architecture. In order to test the feasibility and usefulness of the proposed architecture, a semi-automatic semantic bridging process has been researched and described in Chapter 7.
 - Chapter 8 describes the development efforts concerning the implementation of the MAFRA Toolkit, a fully functional software application that embodies the theoretical research proposed in previous chapters. The applications and experiences performed with the MAFRA Toolkit by third-party projects are described in this chapter.
3. The third part concludes the research and development description:
- Chapter 9 reviews the performed research and achieved results.
 - Chapter 10 describes the ongoing research efforts, together with the envisaged future research.
4. The fourth part includes:
- Annex 1 generically describes the concepts of the relational data model and especially the relational algebra. This annex may be useful for those reading Chapter 6 that are not familiar with this model.
 - Bibliography presents the bibliography used and referred to in this thesis.

Chapter 2

MOTIVATIONS

Managers have perceived the central role the knowledge plays in the capacity of organizations to compete in current socio-economical context [Benjamins *et al.*, 1998]. Knowledge management approaches [Wiig, 2000] aiming to promote responsibilities, expertise, innovation, ideas, participative behavior and social well-being within organizations, are being increasingly adopted.

While these practices are nowadays common in consolidated corporations, a considerable gap still exists in adopting similar approaches in inter-organizations scenarios. Organizations must extend the syntax-based information practice, proceeding to the semantics and pragmatics era [Benjamins *et al.*, 1998], boosting quantity and (especially) quality of the interactions between intervenients. Ultimately, organizations must adhere and implement knowledge-based interoperability mechanisms/solutions.

The “knowledge-based interoperability” expression associates two fundamental characteristics of interactions:

1. Interoperability, seen as the interaction activity performed by two independent entities to achieve a certain goal or state;
2. Knowledge, the “justified belief that increases an entity capacity for effective action” [Nonaka & Takeuchi, 1995].

While information-based interoperability is widespread, knowledge-based interoperability is less common. The main difference between both approaches resides in the fact that unlike information-based interoperability, in which only information is exchanged, in knowledge-based interoperability both the information and its semantics are exchanged between partners. Intervient entities have therefore access to the others entities representation of their “understanding” of the information being exchanged. As such, entities not originally included in certain information community, are able to reason on the semantics and process the information accordingly. However, because knowledge is also composed by information, it is natural that knowledge-based interoperability has something to profit from and provide to the information-interoperability scenarios.

Therefore, the goal of this chapter is to identify scenarios where information and knowledge integration is a fundamental element in the overall system operation. Four conceptual scenarios have been identified, from which some patterns are derived concerning the use of ontologies in knowledge-based interoperability:

- Schema integration;
- Semantic Web;
- Virtual Organizations and E-Commerce;
- Knowledge Management.

Next sections address each of these subjects.

2.1 Schema integration

Schema integration, also referred to as data integration, is the process whereby data is stored in multiple heterogeneous databases and further provided to consumers through a uniform global schema [Beneventano *et al.*, 2001; Halevy, 2001; Sheth & Larson, 1990]. Two distinct but paradigmatic scenarios illustrate the schema integration requirement: database integration and data warehousing. While sharing a few similar characteristics, a few others contribute to a different integration process.

2.1.1 Database integration

Database integration is required once multiple database systems (DBS) are required to provide common uniform view(s) over their data to a user or group of users. Three main dimensions contribute for the characterisation of the database integration problem [Sheth & Larson, 1990]:

- Distribution, concerning the existence of several location for the different database system participating in the integration process;
- Heterogeneity, which concerns the differences between database system components. Differences can be summarized in the data model¹, query language, semantics, and database specific constraints;
- Autonomy, corresponding to the capacity each DBS component has to control its participation in the integrated system.

Variations on the degree of these dimensions in the database integration process lead to distinct database configurations. Furthermore, some dimensions induce other dimensions. For example, the autonomy occurring at design time strongly induces schemas and semantics heterogeneity.

The interface between each DBS component and the rest of the system is supported by several levels of intermediate schemas that progressively abstract the inner details of each DBS component, providing increased syntactic, model and semantic uniformity according to each information community requirements. In distributed, heterogeneous and autonomous DBS, a five-level schema architecture is suggested²:

- Local schema, corresponding to the conceptual schema of each DBS component;
- Component schema is the representation of the local schema in a common data model (CDM)³;
- External schema represents specific perspectives upon the information community schema, considering permissions, constraints and application precise needs of a user or group of users;
- Information community schema represents the overall, accepted schema for/by all DBS components;
- Export schema represents the part of the local schema that is accessible from other DBS.

¹ Data model is the set of grammar and vocabulary defined by a data-modeling paradigm, used to represent the conceptual (i.e. logical) entities and characteristics of the information to be represented in the database. Examples of data models are the relational data model and object-oriented data model. An instantiation of a data model is normally referred to as a database schema.

² While the described perspective concerns the database domain, [Stumme & Maedche, 2001] suggest a similar architecture for the Semantic Web (see 2.2), where ontologies play the role of schemas.

³ CDM is the expression used to denote the role a data model plays in a modeling process, and not the name of a specific data model.

Figure 2.1 (based on [Sheth & Larson, 1990]) illustrates this architecture using the UML notation [UML].

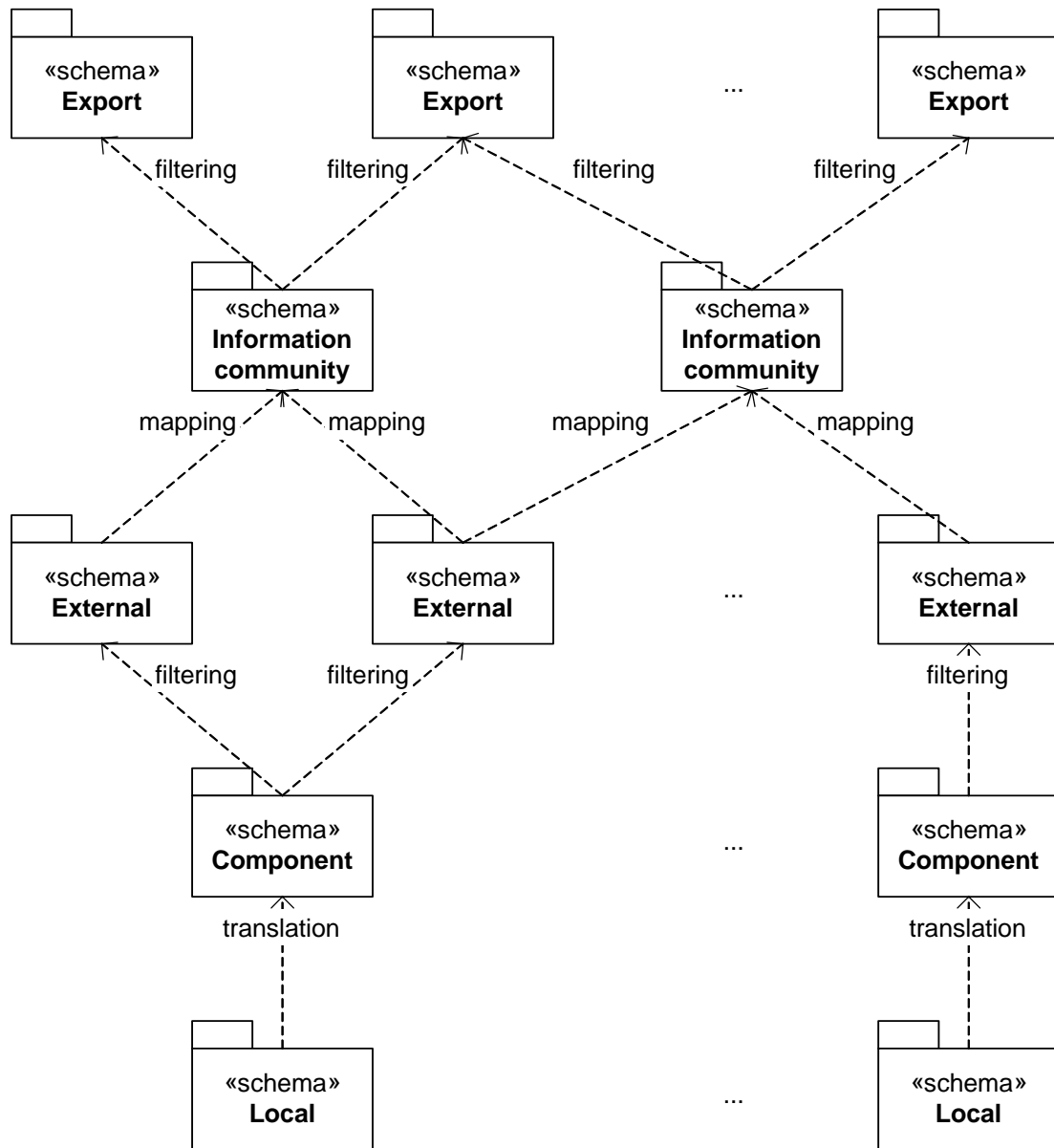


Figure 2.1 - Five-level multi-database integration architecture

Between each implemented pair of schemas, some kind of processing of data is needed. The labels in the lines connecting each schema pair (Figure 2.1) correspond to the most typical processes (operations) on schemas:

- Translation operation employ a source schema (e.g. local schema) specified in a specific data model in the creation of an equivalent target schema (e.g. component schema) specified according to another data model. For example, translating an entity-relation schema into a relational schema;

- Filtering is the operation that creates a target schema (e.g. export schema) from a sub-set of elements from the source schema (e.g. component schema);
- Mapping is the process that specifies functions between elements from a source schema (e.g. external schema) and elements of the target schema (e.g. information community schema) capable to transform data conforming to the source schema into data conforming to the target schema. Mapping is also the name given to the set of functions resulting from the mapping process.

While previous processes occur over schemas, complementary processes occur upon data instances, transforming instances from one schema into instances of another schema. In this sense, the schema-based processes are classified as meta-processes.

Comment 2.1

Because in certain systems the lower schema of the pair fulfils the requirements endorsed to the upper schema of the pair, some of the identified layers are not present. In that case, the lower schema plays both the lower and upper schema roles but no processing is required.

Additionally, in some systems, databases are also responsible for the uniform storage of data arriving from multiple entities. In such cases, the processing occurs in both directions, which implies the specification of inverse equivalence functions when possible.

Despite the three types of processes described in the previous analysis, this work is particularly concerned with the mapping process. The translation and filtering processes are addressed during the rest of the thesis, considering the interdependencies between each and the mapping process. Accordingly, the rest of the analyses focus on the mapping of external schemas into the information schema, i.e. the integration of the information community schema.

Usually, every information community schema relates to multiple external schemas and every external schema may respect to multiple information community schemas. As so, the schemas of the information communities do not exist *a priori*, but result from the combination of several factors and tasks:

- The identification and definition of the elements of the schema required by the community. Thus, unless new external schema of the DBS component is expected, it is useless to define elements that do not exist in the external schemas of the DBS component;
- The identification of the elements of each external schema of the DBS component to integrate in each information community schema in respect to the schema elements of the information community. Thus, it is useless to mention external elements that are not related in the schema of the information community;

- The integration process involves negotiations between the information community and the DBS component administrators about the semantics of related elements. Sometimes it occurs that the export schemas of the DBS component are updated to meet the integration needs;
- The specification of equivalence relations between every element of the schema of the information community and elements of the external schemas of the DBS component.

While the other schema processes can be automated with great success, the schema mapping process is inherently subjective and strongly depends on the information community administrator capabilities to develop and promote a common, uniform schema to the community.

Comment 2.2

Schema integration lacks flexibility to deal with very heterogeneous and autonomous multi-database scenarios [Wache *et al.*, 2001], where information communities are often restricted to one user or application. In such situations, schema integration is very expensive, time-consuming and error-prone.

In such situations, virtual view integration is suggested [Halevy, 2001]. Virtual view integration corresponds to the specification of individual virtual schemas instead of the information community schemas. Views do not respect a common, negotiated viewpoint of the information, but a single dynamic way to access information. Two distinct approaches are possible: over the mediated schema in the local-as-view approach, or over the sources schemas in the global-as-view approach. However, because of its loosely coupled nature, view integration is not suited for databases update but to read-only accesses [Sheth & Larson, 1990].

The evolution and maintenance of the information community schema is a difficult, restricted and time-consuming process. Accordingly, schema integration is not well suited for scenarios where the requirements of the information communities change frequently. Conversely, it better fits the integration of corporate databases than virtual schema integration, because it provides higher levels of coherency and agreement upon the well-established semantics of data and of business process.

Though schema integration is not an all-purpose integration solution, it is well complemented by the view-integration approach. Because the view integration is technologically very similar to the schema integration process, no substantial changes are required in the database integration architecture suggested in 2.1.1. However, it is clear that the responsibilities and administration function are modified, implying changes in the administration paradigm.

2.1.2 Data warehousing

Data warehousing aims to provide a unique, unified repository of data collected from multiple and heterogeneous sources, and to create a materialized uniform view of the data to consumers. Figure 2.2 illustrates the architecture of data warehousing systems according to the architecture presented for database integration in Figure 2.1.

Both approaches share similar data and schema processing tasks, as illustrated by the comparison of Figure 2.1 and Figure 2.2. However, an important difference is noticed. Unlike database integration architecture, where multiple information community schemas are envisaged, in data warehousing

architecture, a unique information community schema exists (named data warehouse schema). As consequence of this particularity, some others characteristics arise and further contribute to the individuality of data warehousing:

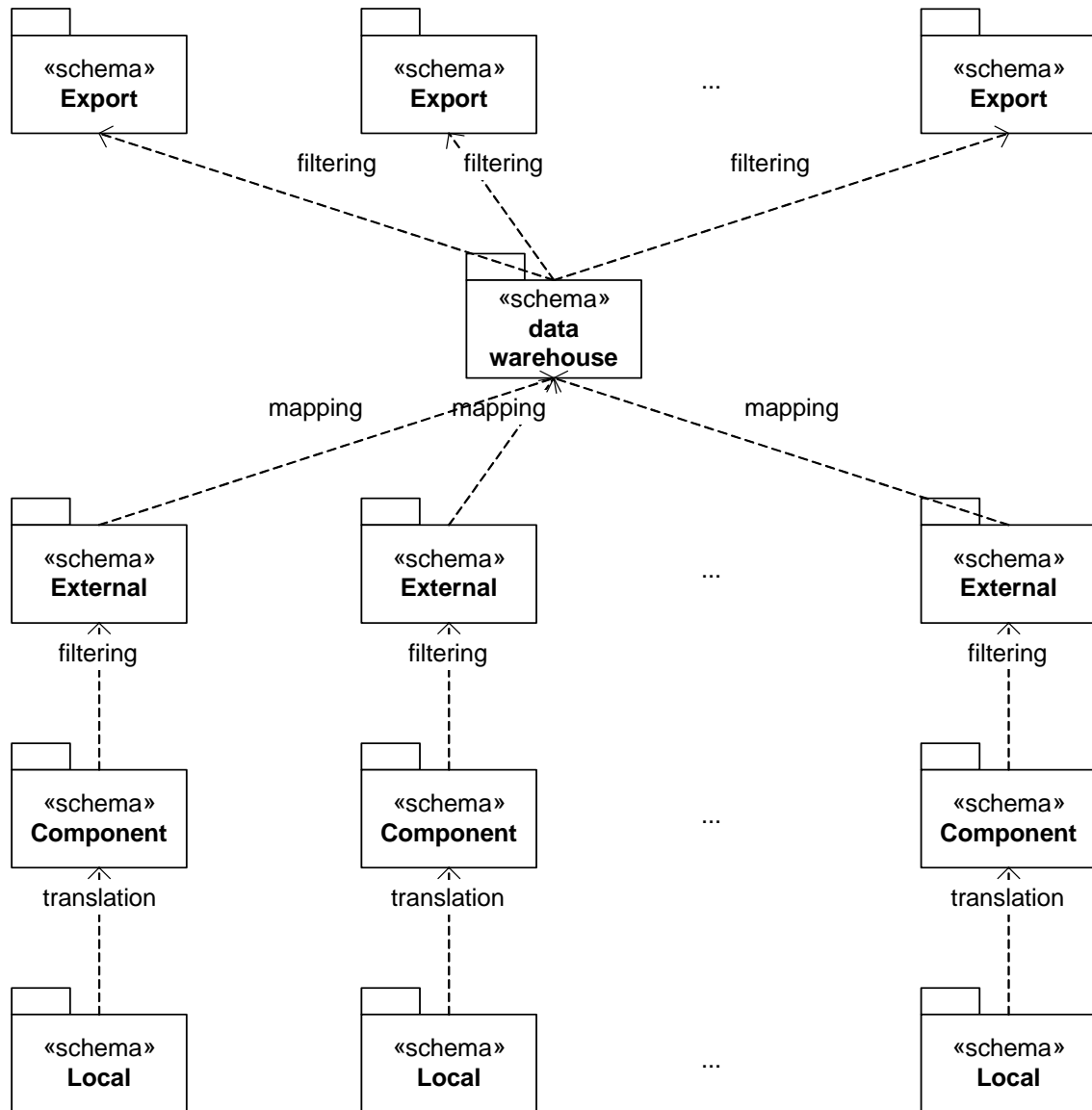


Figure 2.2 – Data warehousing five-level architecture

- Centralization of the data from multiple sources. Unlike database integration whose data is independently distributed over distinct DBS components, data warehousing data is merged in one single repository. Thus, a duplication of data is made into a repository whose schema is (most likely) distinct from the information sources schemas. Centralization provides a unique point of access of all data, even if some of the information sources are unreachable or overloaded at certain moment. Therefore, data in data warehousing is not necessarily accurate in respect to information sources;

- Specialization of the information stored in the repository in comparison to each individual source. Schemas are modeled to better meet the consumers specific requirements, minimizing quantity and consequently, minimizing processing time;
- Extended information. Some processing on data can be materialized into information that did not existed in individual information sources.
- High-performance of the data processing. Three factors contribute to the performance improvement: (i) specialization of data and schemas minimizing processing time, (ii) extended information, which corresponds to the anticipation of consumers requests and (iii) due to centralization of data in one single site, communication delays are minimized.

As observed, the data warehouse integration specific tasks are substantially extended from those described for database integration. In fact, in addition to the mapping processing, six new integration tasks are observed in most data warehouse systems:

- Data merging, corresponding to the duplication of data from data sources in the warehouse repository according to the specific schema;
- Data cleansing, corresponding to the detection of duplicate objects and inconsistencies (different values for the same concept/attribute). Some commercial tools (e.g. Trillium⁴) suggest syntactic, model and semantic transformation as a data cleansing tasks, which correspond to an aggregation of distinct tasks from the mapping and translation processes;
- Computation of extended information, namely aggregation of data into summaries and creation of (extra) indexes;
- Data maintenance, corresponding to reload and process (new or updated) data from information sources. Typically, this process occurs periodically (e.g. per day/week) or is event-driven (e.g. administration request, special query/analysis);
- Detection and removal of expired data;
- Feedback to information sources respecting errors detected during previous processes (e.g. inconsistencies, duplicate objects).

Comment 2.3

As for database integration, view-based integration is possible in data warehousing, but update operations are semantically very sensitive and error-prone. Because data warehouse is a copy of some data from the information sources, queries and updates (when allowed) are executed over the data warehouse repository only. If update operations on information sources are allowed, the system becomes an integrated database system in which the information community schema is unique.

In fact, the data warehousing approach is not well suited for scenarios where update operations are required. Yet, some research has been done on data warehousing,

⁴ <http://www.trillium.com/>

respecting very evolving information sources. In such case, information sources are described according to ontologies and not schemas providing extended descriptions of source repositories [Critchlow *et al.*, 1998].

2.1.3 Data warehousing Vs. Database integration

Data warehousing and distributed database are substantially different. While the first grounds on the notion of centralization, the second grounds on the notion of distribution and autonomy. Both have advantages and disadvantages, especially inherent to this dichotomy.

Data warehousing is a specific implementation of database integration by merging of data, specially developed with data mining and data analysis in mind. However, database integration may profit from data merging too, especially when the merging of the data corresponds to an equivalent business merging. In fact, the solution is very common nowadays as consequence of the increasingly merging events of business corporations.

Database integration architecture typically relies in the use of mediators and wrappers [Wiederhold & Genesereth, 1995] between information sources and information communities. On the other hand, data warehousing traditionally make use of its own very specific (executive oriented) interfaces. Although, as semantic descriptions of information sources are expanded and improved, increased use of automatic mediators is possible and probable [Critchlow *et al.*, 1998].

In common, both approaches require extensive mapping capabilities between schemas.

2.2 Semantic Web

Besides its huge dimension, WWW is a repository of information oriented to human consumption. As it enlarges it becomes increasingly difficult to use and exploit it.

The usefulness of the WWW is mainly a function of three factors:

- The capacity of document classification. Traditionally, “search-engines” classify documents based on their content and metadata. Metadata categories currently employed to characterize documents are implicit to the document type. For example, a HTML document metadata is stated through the METADATA tag, while the Portable Document Format (PDF) provides Subject and Keywords fields. These metadata elements are clearly insufficient, leading to the necessity to mine documents content for complementary information that could support its classification. Increasingly competent tools, using multiple complementary techniques like natural language processing, indexing mechanisms and exploitation of complementary knowledge bases, perform such task. Besides their enormous success, such techniques have important drawbacks when dealing with multimedia documents such as video, pictures and sound. However, independently of extraction mechanisms used, the classification will always

lack the semantic point-of-view of the information author. Therefore, a considerable amount of semantic mismatches, or at least ambiguity, is introduced in the classification;

- The capability to match the user query with the classified documents. Because the query process mainly uses keywords and natural language text, the “search-engine” has to redefine the user query to meet the “search-engine” requirements. Introducing this redefinition, a new (subjective) interpretation occurs in the process, promoting further semantic mismatches or ambiguities;
- The capabilities of the user to process the information. The most common information search methodology exploit web browsers, which support the query phase and the presentation of answers in the computer-device GUI. Further information processing is concerned to the user, which will apply other computer or human-based applications and methods to accomplish his/her own goals.

In 1999, Berners-Lee [Berners-Lee & Fischetti, 1999] suggested the evolution of WWW, shifting from a presentation-oriented paradigm to a computer-aided processing paradigm. The basic idea respects the annotation of the web information with machine-processable description of the information, enabling software entities to mediate between users needs and the information sources [Fensel, 2001]. To model, represent and convey the machine-processable description of the information between information communities ontologies are suggested. Ontologies are made publicly accessible and sharable, allowing information communities to characterize their documents according to the ontologies that best fits the intended semantics of the document content. Conversely, consumers can easily and better match their request against available information. Further detailed description of the ontology concept is presented in Chapter 3.

Once Semantic Web is composed by a large number of very autonomous, (mostly) read-only information providers (databases), Semantic Web can be categorised as a loosely coupled database system [Mitra & Wiederhold, 2001; Popa *et al.*, 2002]. However, current technology is missing the semantic dimension of the information, thus slowing down its adoption by a larger community.

Example 2.1 – Web technology without semantic awareness

UDDI⁵ repositories are used to advertise and discover services in the web but they are not Semantic Web aware. Their use by other services depends on human-coded procedures to determine interoperability components such as services interfaces.

Different research communities such as database [Mitra & Wiederhold, 2001; Popa *et al.*, 2002], artificial intelligence [Decker *et al.*, 1999; Doan *et al.*, 2002; Horrocks, 1998; Sintek & Decker, 2002], knowledge management [Fensel, 2001], E-Business [Fensel, 2001; Sheth & Larson, 1990], agent-

⁵ Universal Description, Discovery and Integration (www.uddi.org) repositories are used to state the functionalities and interface of an entity, so other entities can find it.

based systems [Beneventano *et al.*, 2003; Hefflin *et al.*, 2001; Klusch, 2001] and web services [Ding *et al.*, 2002; Hagel, 2002] are combining efforts and strongly engaged in the Semantic Web vision. In fact, towards Semantic Web environment and technologies, vast research fields are finding new and exciting applications, turning Semantic Web into a fusion of visions.

Considering the limitations identified in the context of current WWW, several developments are therefore necessary in order to promote knowledge interoperability in Semantic Web:

1. Languages and models to represent ontologies, supporting the modelling, representation and sharing of the semantics of the information in different levels of requirements [Critchlow *et al.*, 1998; Gruber, 1993b; Pan & Horrocks, 2003] (see, [Silva, 2002b] for an overview on representation languages for ontologies);
1. Tools to support and promote the annotation of documents according to ontologies [Decker *et al.*, 1999; Handschuh *et al.*, 2002];
2. Reasoning and inference mechanisms to exploit the knowledge sharing in specific contexts [Horrocks, 1998; Motik *et al.*, 2003; Ontobroker];
3. Tools to support and promote the derivation, representation, sharing and maintenance of syntactic and semantic relations between different ontologies [Crubézy & Musen, 2003; Dou *et al.*, 2003; Dou *et al.*, 2002; Maedche *et al.*, 2002b; Omelayenko, 2002b; Park *et al.*, 1998; Silva & Rocha, 2003d; Stuckenschmidt & Wache, 2000];
4. Dissemination of both ontologies and reconciliation relations between them;
5. Development and implementation of Semantic Web aware infrastructures (current implementations lack the semantic dimension).

While some of these requirements are very different from those identified for schema integration, the fourth requirement is indeed much related to the mapping problem described above. Accordingly, the process described to the integration of the information community schema in section 2.1.1 is much valid in this context. However, an important distinction holds between the schema integration and the Semantic Web ontology based integration. In Semantic Web context, the mapping process emergently occurs, especially due to the dynamic, open nature of the environment. Conversely, in the schema integration scenario, users or applications taking part in the process are typically more stable and inter-related. Therefore, the Semantic Web mapping process requires even more machine reasoning and decision support.

Notice that, even if this section has focused on the manipulation of information, Semantic Web goes far beyond this, especially supporting and promoting the subjects of the two following sections.

2.3 Virtual Organizations and E-Business

Virtual organization is a management paradigm aiming to promote cooperation and synergies between autonomous organizations, adopting a cybernetic structure and infrastructure. According to [Silva, 1998]:

“A Virtual Organization is a temporary web of autonomous, cooperative entities, collectively responsible for business activities and through which convey goods and related information.”

Virtual organizations adopt different organizational and technological approaches resulting in different categorization and degrees. Automation and setup time of the formation, duration of the partnership, autonomy, level of integration and type of coordination structure are some of these dimensions. Expressions such extended enterprise, virtual enterprise and supply chain, denotes these variations [Silva, 1998].

The E-Business paradigm aims to promote electronic business interchanges through the Internet in an automatically emergent fashion. E-Business expression normally encompasses both the Business-to-Business (B2B) and Business-To-Customer (B2C) variations. As denoted by name, B2B [Fensel, 2001] suggests inter-enterprise businesses, while B2C respects the business interaction between business organizations and final customers. B2B and B2C services implementations can be complementarily classified as E-Commerce and E-Business services. E-commerce expression applies to any selling or buying service over the Internet, such as e-auctions and e-travel services. E-business expression respects to other type of services such as e-banking, e-learning, health information; though in some stage most of these services include an E-Commerce operation.

While Virtual Organization and B2B/B2C are related concepts, an important distinction arises respecting the goal. The Virtual Organization paradigm focuses on the characteristics of the process to achieve certain goal, which would normally imply the cooperation between multiple autonomous entities. Conversely, typical B2B and B2C implementations aim to reach a business transaction between two partners, even if in some stage of the process third party entities are added to the process to facilitate or accomplish the business process. In common, both have the fact that multiple, heterogeneous autonomous entities participate in goal-oriented conversations through the Internet or other electronic communication meaning.

2.3.1 Agent-based Systems

Despite Virtual Organizations (or business interactions) exploit very distinct approaches, one of the most adopted implementation technology is probably the agent-based paradigm. According to [Silva & Ramos, 1999]

“An Agent is considered an entity capable to interact with others and its environment, sensing and changing it, and according to its own and acquired knowledge, not only react to contextual stimulus but also build and execute action plans to reach its goals”

Hence, agents are characterized according to multiple dimensions like: autonomy, heterogeneity, pro-activity, rationality, sociability and knowledge manipulation capabilities. Agents are therefore well suited to embody the characteristics of Virtual Organization entities.

Notice that agent-based systems and Virtual Organizations are completely different concepts. However, due to their complementarity and characterization interdependency, next considerations are based on the premise that agent-based paradigm will be used to implement Virtual Organizations, and agents stand for and embody the Virtual Organization entities.

Comment 2.4

Agent-based systems are not always knowledge able. Some of them [Bayardo *et al.*, 1997; Klusch, 2001; Silva, 1998] are only information able, which means the semantic component of the knowledge is not explicitly stated but is implicitly encoded in the agents reasoning mechanisms. In such circumstances information is exchanged among agents, while its meaning is (or it is not) implicitly understood by intervenients.

Knowledge able agents are supposed to cope with the difficulties associated with the knowledge, by (1) providing a pro-active information resource discovery, (2) resolving information impedance between consumers and providers, and (3) offering value-added information services and products [Klusch, 2001; Sycara *et al.*, 1998].

Knowledge manipulation technology in agent-based systems results from the combination of characteristics of the agent-based concept. Concerning the knowledge interoperability problem, the following cause-effect considerations arise:

- Because agents are goal-oriented, tasks are distributed among different agents;
- Because agents are autonomous and pro-active entities, the system knowledge is typically distributed among the agents that more directly influence and/or use it;
- Because agents are autonomous, heterogeneous and rational entities, different conceptualizations may be adopted in the system;

- Because agents are socially able, knowledge exchange technology capable to promote and support high-level conversations is needed [Cohen & Levesque, 1995];
- Because agents are potentially heterogeneous, different knowledge representation and exchange technology can be also used.

While this systematization does not intend to be complete or universal, it is sufficiently broader but respects the current problem requirements.

2.3.2 Web services

Web services technology is a very promising technology to support Virtual Organizations and E-Business integration. According to [Hagel, 2002]:

“Web services are business and consumer applications, delivered over the Internet that users can select and combine through almost any device from personal computers to mobile phones.”

Web services are one of the web newest trends, sharing a close and mutual relation with Semantic Web. On the one hand, Semantic Web requires web services to become a reality, and on the other hand, these services make use of the Semantic Web infrastructure and related technology to its implementation [Ding *et al.*, 2002]. Semantic web representation languages (see [Silva, 2002a]) are used not only to represent ontologies but also for the representation of protocols for access web services (e.g. SOAP⁶), description of web services (e.g. WSDL⁷).

Web services are closely related to Virtual Organizations and E-Business in respecting their implementation [Global Exchange Services, 2003]. Web services are in deed envisaged as on of the most promising technology to support inter-organizations interoperability, exploiting the Semantic Web infrastructure and technology, leading E-Business to unprecedented levels. The above-mentioned technologies for description of protocols, characteristics and repositories, provide the basics to automate conversations between disparate service-based computers. Typically, four dimensions contribute for the characterization of web services:

- Service complexity, respecting to the required message and products transactions, protocols, type and number of involved entities. Normally, Virtual Organization and B2B services are much more complex then B2C services;

⁶ Simple Object Access Protocol, <http://www.w3.org/TR/SOAP/>

⁷ Web Service Description Language, <http://www.w3.org/TR/wsdl>

- Security, respecting the messages, values and products transactions. Typically, in order to promote security, mediation entities are involved in transactions, which increase the complexity of the service;
- Negotiation between business entities. Unlike business to customer relations, business to business interactions normally includes discussion and negotiation of very different conditions;
- Information integration between business entities. B2C services do not require integration between business and customer systems, due to both the nature of the interoperability (normally one time or seldom interoperability) and because final customers systems are not typically prepared or with processing and decision capabilities. Instead, B2B services, and especially Virtual Organization, require extensive integration automation to support reliability, accuracy and speed of the process.

2.3.3 Virtual organization and E-Business requirements

Accordingly, in the context of the agent-based systems, and consequently in the adoption of agent-based systems and/in Virtual Organizations, the following knowledge manipulation technologies are deemed necessary:

1. Technology to acquire, represent and exchange semantics of;
 - 1.1 The organizations (web services or agents) information;
 - 1.2 The messages exchanged between organizations;
2. Technology to support heterogeneous syntactic-level interactions, namely translating between different notations;
3. Technology to support heterogeneous paradigm-level interactions, namely translating between different data models;
4. Technology to support heterogeneous semantic-level interactions, namely:
 - 4.1 Supporting and promoting common consensus about knowledge between entities/agents;
 - 4.2 Mapping messages and their content between different knowledge specifications;
5. Workflow approaches capable to automate the formation of cooperative entities network through the exploitation of the semantic description of entities. While workflow approaches are a long-time studied problem, the application of the semantic descriptions of the web services imposes new challenges.

These requirements are clearly similar to those identified for Semantic Web and schema integration scenarios. However, the semantics of the messages are an original element in this scenario. Messages exchanged between interoperability partners are core elements in the conversation protocols, since they carry the meaning and the intended action or attitude upon their contents [Cohen & Levesque, 1995], i.e. the agent (or web service) processing of the information is

dependent on the protocol (the set and the allowed order of the messages), the message (the intended action or attitude) and its content (the information about which the message express an action or attitude).

Example 2.2 – KQML performatives: same content but different meaning

In the following two KQML⁸ messages (performatives), the same information content is differently processed and understood:

```
(stream-all
  :sender agent1
  :receiver agent2
  :language Prolog
  :ontology Travel
  :content "Hotel("Pousada", "4*", "Gerês-Portugal")")
(tell
  :sender agent1
  :receiver agent2
  :in-reply-to id1
  :language Prolog
  :ontology Travel
  :content "Hotel("Pousada", "4*", "Gerês-Portugal")")
```

Through the first message, agent1 asks agent2 for all Hotels with the specified characteristics. The second message is used by agent1 to inform agent2 that a Hotel with the specified characteristics exists.

An eventual semantic clash occurs when certain message is strange in an agent protocol, but a mapping is possible between both agents protocol. For instance, consider that the first message is unknown to agent2. Thus, a mapping between conversations performatives would eventually map between the stream-all message and the ask-all message. Despite the performatives indicate different actions, in some circumstances they can be used interchangeably.

Accordingly, a mapping processing is necessary upon messages, similar to that occurring upon agents (web service) knowledge.

2.4 Knowledge management

Knowledge management (KM) is an interdisciplinary⁹ business model aiming to exploit knowledge towards competitive advantages in business [Wiig, 2000], by sharing expertise, experiences, ideas and responsibilities, by promoting innovation, participative behavior and social well-being within organizations. Knowledge, as understood in the scope of KM is available in all kind of enterprise documents, interactions and activities (e.g. textbooks, multimedia documents, heuristics, skills) even if it is not physically represented.

The technological facet of knowledge management aims to provide methods and technology for:

⁸ Knowledge Query Manipulation Language is used to exchange information messages in agents-based systems.

⁹ http://www.krii.com/downloads/Four_KM_Facets.pdf

- Acquiring, mining and collecting knowledge from all the imaginable sources of information [Benjamins *et al.*, 1998; Fensel, 2001];
- Structuring, organizing and indexing knowledge in order to use it efficiently [Benjamins *et al.*, 1998];
- Maintaining knowledge source [Benjamins *et al.*, 1998; Fensel, 2001];
- Distributing and querying knowledge sources [Benjamins *et al.*, 1998; Fensel, 2001].

One of the most important goals of knowledge management technology is indeed the integration of different sources into a meaningful and useful knowledge base (KB).

Two fundamental observations further arise from the analysis of this scenario:

- KBs are accessed and updated by diverse intelligent entities in the organization. Such entities work on different departments, have different interests, and process the same information in different ways. Hence, different conceptualizations are present in the enterprise. Nowadays, ontologies are one of the most common ways to express conceptualizations;
- WWW, both intranet and internet, is the biggest source of information in the world. However, it is composed by unstructured, incoherent, ambiguous and fuzzy classified documents (see 2.2).

From previous observations, the combination of WWW and ontologies naturally arises, denoting the relevance of Semantic Web initiative to the KM scenario. In fact, Semantic Web and knowledge management initiatives are currently very close and interrelated [Fensel, 2001; Fensel *et al.*, 2003]. On one hand, Semantic Web provides the technology and support for knowledge representation and sharing, and on the other, knowledge management provides proof-based approaches and techniques for acquisition, mining, organization, indexing, querying and distribution of information. Conversely, knowledge management is a great application scenario for the Semantic Web ideas and technology.

Therefore, requirements identified for Semantic Web are in general, also pertinent and valid for KM. Moreover, notice that even respecting the need for fast and accurate information, both approaches are very similar. In fact, even if some Semantic Web applications such as information retrieval require short response time, there are other (e.g. E-Business) in which the accuracy is instead very important, thus, neglecting the response time.

2.5 Outlook

This chapter described four distinct scenarios where ontology mapping is envisaged. The main conclusion arising from this chapter concerns the importance of mapping between data sources for many different problems regarding information integration. In particular, mapping manipulation technology requires the specification of syntax, schematic and semantic relations between distinct

knowledge specifications and their further application in the transformation of data between data repositories. Therefore, it is possible to distinguish and systematize between two fundamental distinct phases in the mapping process: the specification and the transformation phases.

2.5.1 Mapping specification phase

In the mapping specification phase, syntactic schematic and semantic relations are specified between the information models. Two distinct types of operations are necessary:

- Off-line specification. This type of operation occurs when an (eventually) long integration phase is plausible, and a high level of accuracy of the output is necessary. Conversely, the automation and duration of the process are disregarded. Integration and merging of organizations and integration of health-care information are specific application scenarios where off-line specification is required;
- On-line specification. This type of mapping specification occurs when two mutually unknown entities require a rather fast interoperability context. Instead, accuracy of the output is disregarded. The ability to (semi-) automatically map between different information models, discovery and alignment of (web) services are of primordial importance to the success of truly dynamic and autonomous cybernetic business. Information retrieval is a typical application scenario where the speed of the interoperability is preponderant over the accuracy.

Certain application scenarios will use both the on-line and off-line operations modes depending on the specific situation. That is the case of Virtual Organization, E-Business and E-Commerce.

2.5.2 Mapping transformation phase

In the mapping transformation phase, data from repositories is transformed according to the mapping specification. Two distinct types of operations are necessary:

- Pull transformation, or batch transformation, occurs when an (eventually) large repository is to be periodically transformed. Data migration, data merging, data warehouse clean & transform and data model evolution are specific technologies where this approach is necessary;
- Push transformation occurs when a small, limited set of data is to be transformed upon (random) requests. E-business, E-Commerce and information retrieval are envisaged application scenarios for push transformation approach.

The mapping transformation phase is often combined with a translation process, which translates data between different representations languages and data models.

2.5.3 Human intervention

Semantic heterogeneity is the main problem arising in this context, which is above all a consequence of human-based modeling decisions [Sheth & Larson, 1990; Studer *et al.*, 1998]. Ontologies and specially schemas do not carry enough semantics to provide sufficient, correct and impartial reasoning (see 3.2 for a comparison between ontology and schema). Despite incremental extended semantics provided, interpretation of schemas and ontologies is yet an inherently subjective process [Guarino, 1994; Madhavan *et al.*, 2001], constraining application and usefulness of completely automatic mapping processes [Sheth & Larson, 1990]. As consequence, humans play a fundamental role in the process, by detecting, correcting or disambiguating semantic heterogeneities. Instead of neglecting or even reject human being from the process, his presence is considered fundamental.

2.5.4 Common consensus building

However, the participation of human being in the process is not a sufficient condition to avoid semantic clashes between intervenients. In fact, as happens in humans' societies, interoperability often requires a more or less long setup stage (e.g. to determine the language of conversation or accomplish a memorandum of understanding).

In fact, a common characteristic to all scenarios is the need to build consensus between the intervenient entities respecting the mapping specification. The resulting mapping will connect two autonomous entities, which will share knowledge according to such mapping. The existence of a consensus upon knowledge mapping is not a sufficient condition, but at least a necessary condition to the success of interoperability.

2.5.5 Evolution

Repositories and their descriptions tend to evolve in order to meet new requirements, solve problems, and increase performance. As information sources evolve, interoperability process needs to evolve too. Adapting the semantic relations according to the information sources evolution is like the initial specification, a human-oriented task too, and eventually even more time consuming and error-prone.

2.5.6 Semantic Web importance

Semantic web is emerging as a fundamental environment and infrastructure for many different applications. Nowadays, traditional IT domains (e.g. database domain), are converging their technological solutions to those suggested in context of the Semantic Web. Accordingly, it is

recommended that the ideas, proposals and technology arising from this work be aware of Semantic Web research and proposed technology.

2.5.7 Summary of requirements

Based on the systematization of this section, it is now possible to clearly enumerate the technological requirements of mentioned scenarios:

1. Identification, specification and representation of syntactic, schematic and semantic relations between distinct information;
2. Transformation of information exchanged among intervenients according to the specified syntactic, schematic and semantic relations;
3. Negotiation capabilities to reach consensus;
4. Maintenance of the syntactic, schematic and semantic relations;
5. Integration but minimization of the human-being intervention in the mapping process, which suggests the adoption of a semi-automatic, human-supervised ontology mapping system;
6. Semantic web awareness.

Chapter 3

ONTOLOGY

To automate the knowledge-based interoperability, entities are requested to characterize and share their perception of the domain of discourse. Ontologies are envisaged as a convenient concept to support acquisition, representation and exchange of knowledge characterization. Despite no universally accepted definition of the term Ontology exists, two main point-of-views are clearly distinct: ontology as the philosophical discipline and ontology as the knowledge engineering artifact.

As philosophical discipline defined by Aristotle, Ontology analyzes the nature and organization of reality, “all the species of being *qua* being and the attributes which belong to it *qua* being” (Aristotle, *Metaphysics*, IV, 1). This perspective is not significant for this work and therefore it is no longer analyzed.

Instead, the understanding upon ontology in the context of knowledge engineering is very important for this work. The most cited definition of ontology has been suggested by Gruber in 1993 [Gruber, 1993a] as “an explicit specification of a conceptualization”. Later in 1998, Studer and colleagues [Studer *et al.*, 1998] extended this definition to “Ontology is a formal, explicit

specification of a shared conceptualization”. “Formal” refers to the fact that the ontology should be machine-readable, excluding therefore the natural language based ontologies. ‘Shared’ reflects the notion that ontology captures consensual knowledge, which means that it is not private to some individual, but accepted by a group, referred as information community. “Specification” assumes an embodied existence using a specific construction artifact like an ontology representation language or natural language. It is said “explicit” because ontology entities are clearly distinguished and interrelated. It refers to a “conceptualization” in the sense that it refers to an abstract, cognitive model of some domain, which identifies the domain concepts and their characteristics.

However, according to Guarino and Giaretta [Guarino & Giaretta, 1995], ontology is a “logical theory which gives an explicit, partial account of a conceptualization”. This interpretation relies on the notion of ontological theory, referred as “a set of formulas intended to be always true according to a certain conceptualization”. This perspective additionally states that:

1. Ontologies are special kinds of knowledge bases;
2. Any ontology has its underlying conceptualization;
3. The same conceptualization may underlie different ontologies;
4. Two different knowledge bases may commit to the same ontology.

Combining previous three convincing perspectives, the following definition has been drawn:

“Ontology is a formal, partial and explicit specification of a shared conceptualization.”

However, besides the fact that previous ontology definition include multiple features required in this context, the definition is extremely vague, allowing multiple interpretations and implementations. Thus, a deeper analysis of ontology is necessary, addressing both conceptual and implementation-dependent characteristics.

3.1 Characterization of ontologies

Since no common definition of ontology exists, it is natural that no consensus exists concerning the characterization of ontology too. However, analysis and synthesis of literature allows the identification of some characterization patterns that provide a basic framework for their characterization.

3.1.1 Generality

Generality is the quality or state of being general. The more generic the ontology is the more entities might understand the characterization of the elements of the universe it describes. Specificity is the inverse dimension: the more specific the ontology is, less entities accept or commit

to that characterization. This dimension is particularly related to the reusability feature of the ontology.

Guarino [Guarino, 1997a] identifies four types of ontologies according to generality:

1. Top ontologies, which describe extremely generic concepts such space, time [Santos & Staab, 2003], events and roles, independent of domain or application. CYC¹⁰, Pangloss¹¹ and Mikrokosmos¹²; are some examples of top ontologies;
2. Domain ontologies describe specific elements of specific domain of discourse like health, electronics and planning [Planserve, 2003]. This type of ontologies specializes or make use of concepts defined in top ontologies;
3. Task ontologies describe conceptual elements related to generic tasks or actions, such as quality control, maintenance, planning and scheduling. These ontologies specialize or make use of elements defined in top ontologies;
4. Application ontologies recur to the combination of both domain and task ontologies to describe specific elements and tasks of a domain. Elements in top ontologies can also be used in case the application ontologies extend application and task ontologies.

Towards the characterization of Problem Solving Methods (PSM), Studer and colleagues [Studer *et al.*, 1998] suggest the categorization of ontologies as:

- Generic ontologies, corresponding to the top categorization of ontologies;
- Domain ontologies, corresponding to the aggregation of domain and task categories of Guarino. The Enterprise Ontology [Uschold *et al.*, 1998] and TOVE [Fox & Gruninger, 1997] in the domain of the enterprise modeling are two examples of this development approach;
- Application ontologies, corresponding to the application category of Guarino
- Representation ontologies, which correspond to models upon which the other categories can be modeled and represented (in [Silva, 2002a] a characterization of different ontology representation models is provided).

3.1.2 Granularity

Granularity dimension respects the capacity of conceptualization at different levels of abstraction [Uschold & Jasper, 1999; Studer *et al.*, 1998], or the level of detail or precision [Uschold & Jasper,

¹⁰ Lenat, D. B., Guha, R. V.; “Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project”; Addison- Wesley Publishing Company, Inc.; CA, EUA; 1990.

¹¹ Knight, K. and Luk, S.; “Building a Large Knowledge Base for Machine Translation”; in Proceedings of American Association of Artificial Intelligence Conference (AAAI-94); Seattle, WA, EUA; 1994.

¹² Mahesh, K.; “Ontology Development for Machine Translation: Ideology and Methodology”; New Mexico State University, Computing Research Laboratory MCCS-96-292; 1996.

1999] in which the universe of discourse is modeled. Granularity is classified as thin (fine) or coarse. The coarse grain ontologies are less detailed (and more abstract) than fine grain ontologies, and vice-versa. Coarse grain ontologies are normally developed with the specialization approach in mind: coarse (abstract) grain elements are further specialized into finer grain (detailed) elements.

Granularity is particularly aware of the type of the application. Guarino [Guarino, 1997b] identifies two distinct opposite types:

- On-line ontology is a coarse grain ontology applicable in scenarios in which accuracy of the systems is disregarded instead of the responsiveness and usability of the system;
- Off-line ontology is fine grain ontology, applicable in scenarios where high-level accuracy is required. Fine grain ontologies typically require more computational efforts than coarse grain ontologies.

3.1.3 Formality

Formality is the ontology characteristic that measures the conformity to conventional rules, preventing interpretation ambiguities. Accordingly, formal ontologies are particularly suited in high-level accuracy and machine-based scenarios. This dimension is partially inherited from the ontology representation language. In fact, the formality of the ontology is constrained by the representation technology but the modeling and development phases may induce ambiguities too.

Though many formality-based classification exist [Studer *et al.*, 1998; Uschold & Jasper, 1999; Stuckenschmidt *et al.*, 2000], according to Uschold and Jasper [Uschold & Jasper, 1999] four types are clearly distinct:

- Informal ontologies are typically represented through natural language texts and glossaries. This type of ontology is very useful in knowledge acquisition and negotiation phases, providing the means so domain experts, and other users, lacking ontological engineering capabilities, participate and share their viewpoints during the knowledge-based system development process. Texts and glossaries are very expressive but very ambiguous;
- Structurally informal ontologies are both human and machine readable, but their interpretation is still very ambiguous. This type of ontologies is very useful in the early stages of the knowledge-based system modeling. In these phases, it is important to systematize knowledge and achieve consensus between different perspectives. Structurally informal ontologies such as taxonomies, provide limited but useful machine-based validation of knowledge in these phases;
- Semi-formal ontologies are the most common type of on-line ontologies since human and machine processing is possible and the typical computational effort required are limited. Description logics and frame-based languages are the most common languages used to represent

this type of ontologies. Semi-formal ontologies are fast becoming the most common type of ontologies, especially because they are very popular in the semantic web;

- Formal ontologies are very powerful representations, providing their formal verification and univocal utilization. However, the computational effort and time needed to process them are typically not deterministic. First order logic based languages are commonly used to represent these ontologies.

Notice that, in many cases, ontologies serve to describe the knowledge-based system knowledge. Consequently, it is common that less formal ontologies are used towards the specification of more formal ones.

The previously proposed definition of ontology includes neither the informal nor the structurally informal ontologies types. This is not a limitation of the definition though, but a mandatory requirement for the rest of the work: only formal and semi-formal ontologies are considered in the ontology mapping process.

3.1.4 Roles

This section tries to describe some of the roles ontologies play in knowledge engineering¹³. This is not a characteristic of the ontology in itself, but depends foremost on its application. According to literature [Stuckenschmidt *et al.*, 2000; Studer *et al.*, 1998; Uschold & Jasper, 1999; Gruninger & Fox, 1995; Uschold *et al.*, 1998] ontology roles can be systematized and synthesized into:

- System modeling artifact. Ontologies provides relevant mechanisms for acquisition, building consensus, representation and exchange of characteristics of the domain between different intervenients in the process and supporting, since early stages, the formal specification and validation of perspectives;
- Interoperability artifact, between persons, enterprises and systems. Through ontologies, heterogeneous entities are able to share their characterization of their universe, providing the basic support to the correct and univocal interpretation of exchanged contents.

Depending on the roles ontologies play in the system, different characteristics are observed. Yet, ontologies are often used in multiple roles. In fact, it is common that (formal) ontologies applied in representation of knowledge characteristics within systems are used in the interoperability between systems too.

¹³ Knowledge engineering is the process of building Knowledge-Based Systems, which in turn are systems that apply and exploit knowledge about some domain to solve a problem from that domain.

3.1.5 Miscellaneous characteristics

Several characteristics are less referred in literature but acquire special relevance in the scope of this work:

- Modularity concerns the separation of the ontological description upon distinct ontologies. Modular ontologies would provide an interface that is kept unchanged for long time, supporting dependency relations with other ontologies;
- Dependency concerns the relations between ontologies. Specialization (related to the generality dimension), extension (expanding the characterized domain of discourse) and referencing (refer to some element or part of other ontology) are three types of dependency relations. Notice that normally the dependency relation is unilateral (i.e. only one of the related ontologies is aware of the dependency). Problems arise when the non-aware ontology is modified in ways that affect the other ontology. Maintenance and evolution mechanisms are necessary;
- Size of the ontology depends on the modularity and dependencies. On one side, ontology size must be kept as small as possible, through modularity and dependency relations. On the other side, dependency problems arise easily in distributed and uncontrolled environments such as semantic web. A trade-off between size, modularity and dependencies should be carefully considered.

These are some of the most important characteristics of ontologies. Others, like autonomy, paradigm/model used and evolution are also very important characteristics, but because they have been referred during prior descriptions, no further description is considered necessary.

3.2 Ontology Vs. Database schema

One of the greatest controversies arising upon the definition of ontology is related to its distinction with database schema. Though no extensive debate is intended, a short comparison would be useful to determine the pertinence of both concepts in scope of mapping processing requirements.

In database context, schema is a “formal architecture for a database”¹⁴ or “the organization or structure for a database”¹⁵. In the XML user community, “schema is a model for describing the structure of information”¹⁶.

Accordingly, schemas are concerned with the organization and structure of data, while ontologies are concerned with the identification and specification of the meaning of the data. Schema objects

¹⁴ http://mediagods.com/glossary/What_is_a_schema.html

¹⁵ http://iroi.seu.edu.cn/books/ee_dic/whatis/schema.htm

¹⁶ <http://www.xml.com/pub/a/1999/07/schemas/whatis.html>

are the logical structures that directly refer to the database data, such as tables, views, hierarchy of concepts, sequences, stored procedures, synonyms, indexes, clusters, and database links¹⁷. Instead, ontologies elements capture the meaning of the data in a way that multiple heterogeneous entities can reason and simultaneously understand the subjacent data and its relations with the world entities.

Several other characteristics contribute to distinguish between both concepts, but most of them are not conceptual characteristics but implementation dependent.

A typical, pragmatic manner to relate both concepts is to understand ontology as an extension of a schema, in the sense that both organize and structure data, but ontologies (and their representation languages) are envisaged as more powerful in representing semantics than database schemas. Semantics specification suggests the use of axioms capable to constrain and describe knowledge (e.g. inverse, equivalent, transitive properties, quantifiers, Description Logic-based constraints). However, schemas definition languages (SDL) rarely support semantic axioms. Still, two observations arise:

- On one hand, an increasingly number of (schema) data models provides semantic axioms;
- On the other hand, semantic axioms normally found on (implemented) ontologies are not very expressive and can be found in schemas too.

A very powerful mechanism available in some of the most recent ontology representation languages is the capability to modularize and define dependencies between ontologies. Behind this mechanism is a very simple idea: a web of semantic descriptions of the world(s) that can be increasingly extended, refined and interrelated (see 3.1.5). This web of ontologies would increasingly form a foundation for automatic interrelation and reasoning upon knowledge.

This modeling approach can be found in most of the semantic web oriented ontology representation languages, such as OWL, DAML, OIL, DAML+OIL and RDFS¹⁸. Instead, this modeling approach is not very common in SDL's and even less commonly used in schema implementations.

Moreover, most of the ontology representation languages in the context of semantic web adopt an object-oriented modeling approach, promoting the notion of taxonomy (hierarchy) of concepts. While the hierarchical model has been extensively used in modeling databases two decades ago, the object-oriented database management systems are becoming increasingly popular and reliable.

¹⁷ <http://members.tripod.com/mdameryk/OrclOverview.htm>

¹⁸ See [Silva, 2002a] for a brief characterization and comparison between these and other representation languages.

A lexical layer is also commonly referred as a distinctive characteristic. Normally a lexical characterization of ontology elements (synonyms, different languages, antonyms, homonyms, holonyms, etc.) is suggested. However, this layer is not mandatory, and evidences show that its presence on ontologies heavily depends on the intended roles and implementation. Further, most of the database development paradigms and DBMS promote this layer through artifacts such as data dictionaries and association of metadata.

3.2.1 Overview of informal characteristics

Two conclusions arise from previous debate:

- Conceptual perspective shows that ontology and schema are completely different concepts: schema is concerned with structure and organization of data, while ontology is concerned with the description and meaning of the data;
- Implementation reality shows that ontologies and (database) schemas are often very similar, and differences arise mostly as a question of degree.

According to the last perspective, Table 3.1 presents a tabled comparison between ontology and schema. The first part of the table synthesizes the differences just presented, while the second part distinguishes both terms according to the dimensions identified in 3.1. Complementarily, some facts are also included, such as data models and representation and manipulation languages.

Table 3.1 – Comparison between Ontology and Schema

Characteristics	Ontology	Schema
Data types	Not mandatory	Mandatory
Structure	Present	Present
Lexical layer	Suggested	Not mandatory
Semantic axioms	Suggested	Uncommon
Axioms expressivity	Extended logic based constraints	Poor [Sheth & Larson, 1990]: Cardinality
Model	OO, Property-centric, Frame-based, Description Logics	Relational, OO, ER, Hierarchical
Representation languages	OWL, DAML+OIL, RDFS, XOL	SQL, XML Schema, DTD
Query languages	RDF Query, RQL, KIF	SQL, CODASYL, XQuery
Generality	Depends on implementation	Depends on implementation
Granularity	Depends on implementation	Depends on implementation
Formality	Formal or semi-formal	Formal or semi-formal
Role	Modeling and interoperability	Modeling
Modularity	Very common	Common
Dependency	Very common	Uncommon

Due to the implementation-based evaluation, previous comparison has limited usefulness. In order to achieve the goals of this comparison, some assumptions are necessary respecting the ontology and schema characteristics.

Rather than absolutely quantify envisaged characteristics, a relative qualification is suggested respecting the requirements identified in 2.5.7. For each requirement, both the conceptual and most common characteristics of implemented artifacts are considered, thus resulting in a subjective qualification. Table 3.2 synthesizes the qualification.

Table 3.2 – Ontology and schema abilities to support the requirements

Requirements	Ontology	Schema
1. Identification, specification, representation and maintenance of relations between distinct information sources		
1.1 Syntactic relations	+	+
1.2 Schematic relations	+	+
1.3 Semantic relations	+	-
2. Features supporting the negotiation to reach consensus	+	-
3. Transformation of information exchanged among intervenients according to the specified relations		
3.1 Syntactic relations	+	+
3.2 Schematic relations	+	+
3.3 Semantic relations	+	-
4. Maintenance of the relations		
4.1 Syntactic relations	+	+
4.2 Schematic relations	+	+
4.3 Semantic relations	+	-
5. Integration but minimization of the human intervention in the mapping process	+	-
6. Semantic Web-awareness	+	+

As a conclusion, even if both concepts convey similar elements, it is perceived that ontologies better fulfill and support the identified requirements (2.5.7). In particular, the automation of the process will strongly benefit from the extensive availability of semantically rich ontologies.

Finally, even if ontology and database schema may share some characteristics, they are conceptually different and therefore require distinct development models.

3.3 Formal Definition of Ontology

According to previous description, it is now possible and advisable to introduce a formal definition of ontology, or in other words, define an ontology model.

In context of this thesis, ontology comprehends three distinct layers:

- The schematic (or structural) layer, which specifies domain and/or application entities, their inter-relations (e.g. subclass of) and properties;
- The lexical layer that characterizes entities and their properties with natural language lexicons, giving them a real-world meaning (e.g. XPTO entity corresponds to real world entity Person or Individual);
- The axiomatic layer, which constrains the interpretation and application of entities through axioms or rules (e.g. parents of an instance of Person are instances of Person).

From these observations, an adaptation of the Motik and colleagues formal definition of ontology [Motik *et al.*, 2003] is adopted.

3.3.1 Schematic layer

Schematically, ontology is a tuple in the form of:

$$\mathcal{O} := (\mathcal{C}, is_a, \mathcal{P}, \sigma)$$

where:

- \mathcal{C} is the set whose elements are called concepts (or classes) defined by the domain expert;
- is_a is the partial order on \mathcal{C} representing the hierarchical relation between concepts, which is commonly referred as “subclass of” relation:

$$is_a \subseteq \mathcal{C} \times \mathcal{C}$$

is_a is a reflexive, transitive and anti-symmetric relation, which is normally represented extensionally, either as:

- A binary relation in the form of $Concept_2 is_a Concept_1$;
- A set of ordered pairs in the form of $(Concept_2, Concept_1)$;
- A binary predicate (truth-valued function) in the form of $is_a(Concept_2, Concept_1)$.

Example 3.1 – Hierarchical relation in an ontology

The hierarchical relation “Student is subclass of Person” over the set of concepts $\{Student, Person, Car\}$, are defined either as (i) $is_a_1 = \{(Student, Person)\}$, (ii) $Student is_a Person$ or (iii) $is_a(Student, Person)$

Two complementary functions are available, providing access to the relative roles of concept in the relation:

- $subConceptOf : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ gives the set of concepts the concept is sub-concept of;
- $superConceptOf : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ gives the set of concepts the concept is super-concept of.

- \mathcal{P} is the set whose elements are called properties;
- σ is a function which assigns to every property their domain and range concepts:

$$\sigma: \mathcal{P} \rightarrow \left(2^{\mathcal{C}} \setminus \{\emptyset\}\right) \times \left(2^{(\mathcal{C} \cup \{Literal\})} \setminus \{\emptyset\}\right)^{19}$$

This function is normally represented extensionally, as referred for the *is_a* relation. Two complementary functions are available:

- $domain: \mathcal{P} \rightarrow 2^{\mathcal{C}} \setminus \{\emptyset\}$ gives the set of domain concepts of the property;
- $range: \mathcal{P} \rightarrow 2^{\mathcal{C} \cup \{Literal\}} \setminus \{\emptyset\}$ gives the set of range concepts of the property.

Furthermore, properties are further characterized according to their range:

- The property is said to be an Attribute, when $\forall p \in \mathcal{P} \quad Literal \in range(p)$;
- The property is said to be a Relation, when $\forall p \in \mathcal{P}, \exists c \in \mathcal{C} \quad c \in range(p)$. The set of all relations of an ontology is referred by \mathcal{R} .

Despite it is not explicitly referred in the ontology definition, ontology entities are the elements of the domain of discourse, and are therefore the union of Concepts, Properties and Literal:

$$\mathcal{E} := \mathcal{C} \cup \mathcal{P} \cup \{Literal\}$$

Accordingly, it is very easy to make a parallelism between the structural elements of ontology model and the relational data model (see Annex 1). In particular, the σ function structures elements such as the relations data model relations (tables), in which the domain concept is the name of the table and the range is the name of the attribute of the table.

3.3.2 Lexical layer

Usually, ontology entities names are meaningful, i.e. their name corresponds, in some natural language, to the represented concept or property. However, it also occurs that either by decision or by necessity, names are meaningless (also referred as opaque). A middle term occurs though, i.e. the names are meaningful but not enough to be used either by human-being or machines. In such cases, it is possible to associate lexical elements to the ontology entities, promoting the comprehension to humans and machines. Such lexical entities are typically real-world lexicons, and are associated with ontologies entities through the lexical layer of the ontology.

¹⁹ Literal is the specific ontology representation language concept, responsible for encoding instances of primitive types, such as strings and numbers. Instances of type Literal are also referred as constants.

The lexical layer of an ontology is formally defined by a tuple in the form of:

$$\mathcal{L} := \{\mathcal{L}^C, \mathcal{L}^P, \alpha^C, \alpha^P\}$$

where:

- \mathcal{L}^C and \mathcal{L}^P are sets of entities named lexical entries, that correspond to real-word terms for the concept and properties respectively;
- α^C and α^P are the relations that associate lexical entries with concept and properties respectively.

While these relations are generically equivalent to the linguistic synonym relation, other relations might be defined, such as localization relations, capable to associate lexicons of different languages (e.g. Portuguese, English, and German). However, during this thesis such requirement will not be used and therefore is no longer described.

Accordingly, the formal definition of ontology becomes:

$$\mathcal{O} := (\mathcal{C}, is_a, \mathcal{P}, \sigma, \mathcal{L})$$

3.3.3 Axiomatic layer

Conceptually, the axiomatic layer is a set of ontology axioms expressed in an appropriated logical language (e.g. first order logic). Alternatively though, the ontology representation language may define specific modeling constructs to substitute the typically wide expressivity of logical languages. Due to their well-established nomenclature and semantics, these modeling constructs are widely understandable and accepted, motivating its application instead of general-purpose logical statements. Property constraints like transitivity, inversion and cardinality are some of the most commonly provided constraints.

Example 3.2 – Ontology axioms in the form of modeling constructs

Consider the following relations as examples of axiom modeling constructs:

- *MaxCardinality*, which determines the maximum number of instances of a property. It is represented as a relations in the form of:

$$MaxCardinality : \mathcal{P} \rightarrow \mathbb{N}^+$$

- *InverseRelation*, which defines that two relations are mutually inverse. For instance, the “is parent of” relation is inverse of “is son of” relation. The *InverseRelation* axiom might be defined as the relation:

$$InverseRelation : \mathcal{R} \rightarrow \mathcal{R}$$

Besides these, in the context of semantic web, DAML, DAML+OIL and OWL ontology representation languages adopted other types of axioms derived from the Description Logic (DL) approach of modeling, serving as inference rules²⁰.

Example 3.3 - Concept inference based on axioms

Consider the following very simple ontology:

$$\mathcal{O}_1 = (\mathcal{C}_1, is_a_1, \mathcal{P}_1, \sigma_1)$$

$$\mathcal{C}_1 = \{Person\}$$

$$is_a_1 = \{ \}$$

$$\mathcal{P}_1 = \{has_gender\}$$

$$\sigma_1 = \{has_gender\{Person, Literal\}\}$$

One can additionally specify the class Woman according to the Person class:

$$is_a(Woman, Person) \Leftrightarrow Person.gender == "feminine"$$

The following inference rule is automatically derived from previous definition by the logical reasoner:

$$\forall x \in \text{instanceOf}(Person) \quad x.gender == "feminine" \Rightarrow x \in \text{instanceOf}(Woman)$$

Therefore, it is possible to infer that any instance of Person whose gender is “feminine”, might also be (or is better) categorized as Woman.

Ontology definition becomes therefore a tuple in the form of:

$$\mathcal{O} := (\mathcal{C}, is_a, \mathcal{P}, \sigma, \mathcal{L}, \mathcal{A})$$

In nowadays ontologies, the axiomatic layer is often absent. This is often due to the ontology representation language but also to modeling decisions.

3.3.4 Formal definition of Knowledge Base

In scope of this work, Knowledge Base (KB) is an instantiated ontology, also known as populated ontology [Kalfoglou & Schorlemmer, 2003]. Knowledge base is formally defined as the tuple:

$$\mathcal{KB} := (\mathcal{O}, \mathcal{I}, inst\mathcal{C}, inst\mathcal{P})$$

where:

- \mathcal{O} is an ontology;
- \mathcal{I} is a set of elements called instances (or objects), corresponding to the instantiation of $\mathcal{C} \cup \{Literal\}$;

²⁰ Inference rules are applied to both (i) describe ontological entities based on previously defined ontological entities and (ii) determine what additional facts can be implied if other facts are known.

- $inst\mathcal{C}$ is the function that associates ontology concepts with the set of respective instances:

$$inst\mathcal{C} : \mathcal{C} \rightarrow 2^{\mathcal{I}}$$

$inst\mathcal{C}(C) = I$ is equivalent to $C(I)$.

- $inst\mathcal{P}$ is the relation that associates pairs of instances through ontology properties (referred as property instance), corresponding to a relationship between the two concept instances:

$$inst\mathcal{P} : \mathcal{P} \rightarrow 2^{\mathcal{I}} \times 2^{\mathcal{I}}$$

$inst\mathcal{P}(P) = (I_1, I_2)$ is equivalent to $P(I_1, I_2)$.

Accordingly, an ontology instance is any data element represented according to (and coherent with) the domain ontology.

3.3.5 Example 3.4 - Simple ontology and knowledge base

The following definition corresponds to the specification of an ontology and knowledge base:

$$\begin{aligned} \mathcal{KB}_1 &= (\mathcal{O}_1, \mathcal{I}_1, inst\mathcal{C}_1, inst\mathcal{P}_1) \\ \mathcal{O}_1 &= (\mathcal{C}_1, is_a_1, \mathcal{P}_1, \sigma_1, \mathcal{L}_1, \mathcal{A}_1) \\ \mathcal{C}_1 &= \{Researcher, Institution\} \\ is_a_1 &= \{ \} \\ \mathcal{P}_1 &= \{hasName, researchesIn, inCity\} \\ \sigma_1 &= \left\{ \begin{array}{l} hasName(Researcher, Literal), hasName(Institution, Literal), \\ researchesIn(Researcher, Institution), \\ inCity(Institution, Literal) \end{array} \right\} \\ \mathcal{I}_1 &= \left\{ \begin{array}{l} i_1, i_2, i_3, i_4, "Nuno Silva", "João Rocha", "GECAD-ISEP", \\ "WIM-FZI", "Porto", "Karlsruhe" \end{array} \right\} \\ inst\mathcal{C}_1 &= \left\{ \begin{array}{l} Researcher(i_1), Researcher(i_2), Institution(i_3), Institution(i_4), \\ Literal("Nuno Silva"), Literal("João Rocha"), \\ Literal("FZI-WIM"), Literal("GECAD-ISEP"), \\ Literal("Porto"), Literal("Karlsruhe"), \end{array} \right\} \\ inst\mathcal{P}_1 &= \left\{ \begin{array}{l} hasName(i_1, "João Rocha"), hasName(i_2, "Nuno Silva"), \\ hasName(i_3, "GECAD-ISEP"), hasName(i_4, "WIM-FZI"), \\ researchesIn(i_1, i_3), researchesIn(i_2, i_3), researchesIn(i_2, i_4), \\ inCity(i_3, "Porto"), inCity(i_4, "Karlsruhe") \end{array} \right\} \end{aligned}$$

Using the UML notation, the previous knowledge base is represented according to Figure 3.1:

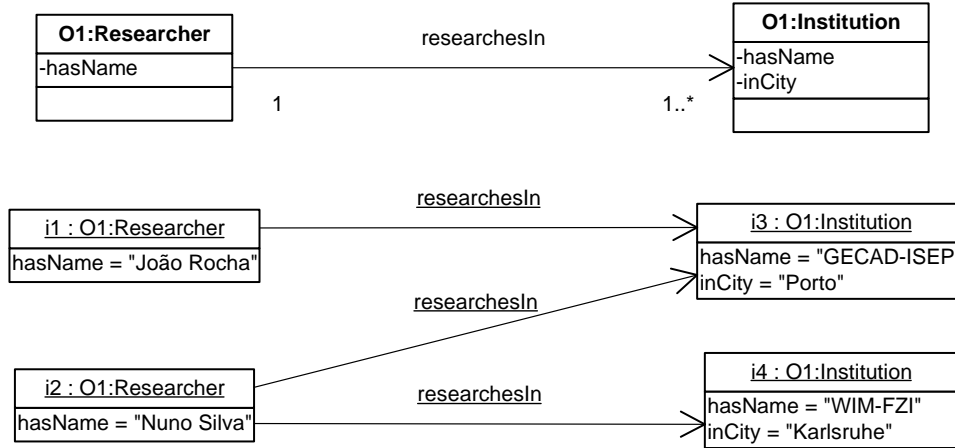


Figure 3.1 – UML representation of the previous ontology and knowledge base

Concerning the lexical layer of the ontology, one might defined it as:

$$\mathcal{L}_1 = \{\mathcal{L}_1^C, \mathcal{L}_1^P, \alpha_1^C, \alpha_1^P\}$$

$$\mathcal{L}_1^C = \{"member", "research\ group"\}$$

$$\mathcal{L}_1^P = \{"name", "memberOf"\}$$

$$\alpha_1^C = \{("member", Researcher), ("research\ group", Institution)\}$$

$$\alpha_1^P = \{("name", hasName), ("memberOf", researchesIn)\}$$

Concerning the axiomatic layer definition, in case $\mathcal{P}_1 = \mathcal{P}_1 \cup \{employs\}$ and

$\sigma_1 = \sigma_1 \cup \{employs(Institution, Researcher)\}$, one might define the following ontology axioms:

$$\mathcal{A}_1 = \{MaxCardinality(hasName, 1), InverseRelation(researchesIn, employs)\}$$

According to the last axiom it is possible to infer the following properties instances, which were not present in the previous knowledge base:

$$inst\mathcal{P}_{inferred} = \{employs(i_3, i_1), employs(i_3, i_2), employs(i_4, i_2)\}$$

3.4 Summary

This chapter presented, described and analyzed the concept of ontology. Although the provided characterization focuses on the use of ontology in the knowledge engineering field, no universal definition exists. Yet, considering the pragmatic use of ontologies envisaged in the scope of this thesis, a particular definition is presented.

However, even if the introduced specific definition is too generic to be used in context of this thesis, it motivated the comparison of ontology with the concept of database schema. Later, as last step through the univocal interpretation of ontology artifact, an ontology model has been defined.

SECOND PART

Chapter 4

ONTOLOGY MAPPING FRAMEWORK

This chapter describes the MAFRA – MApping FRAMework. MAFRA has been specified in the scope of this thesis, and first introduced in [Maedche *et al.*, 2002b], and further characterized and proposed to other communities in [Maedche *et al.*, 2002a; Silva & Rocha, 2003a].

MAFRA has been developed with three goals in mind:

- The analysis and systematization of the ontology mapping process according to the motivational scenarios and the requirements derived in Chapter 2;
- Create a framework for the classification of projects, approaches and technology related with the ontology mapping process. In early stages of the study, several problems have been faced to determine the scope and pertinence of distinct research works to this thesis, specially due to the multiple, ambiguous and incoherent terminology used in the domain;
- Develop a state-of-the-art description of the ontology mapping research field, as a foundation for further efforts.

MAFRA is the first attempt ever made to describe the overall ontology mapping process. No literature has found concerning any of the goals just enumerated.

4.1 Quality vectors

Before any attempt to described and analyze the ontology mapping process, it is important to define some quality vectors. Combining previously mentioned requirements (2.5.7) with the empirical and common sense knowledge assets, seven quality vectors have been derived [Silva & Rocha, 2003a]:

1. Applicability, which concerns with the type of mapping relations that are supported by the system;
2. Semantic Expressivity, which concerns with the capacity of the system to explicitly express the semantic relations;
3. Automation, which concerns with the support the ontology mapping system is able to provide to the human being;
4. Modularization, which concerns with the system characteristic to be build upon the combination of small, simple modules into a more complex whole;
5. Reutilization of components, which concerns with the exploitation and application of knowledge created from previous ontology mapping experiences and its recycling when obsolete;
6. Declarativity, which concerns with the capabilities of the system to supply conditions so the domain expert focuses on the semantics (what to) instead of the syntax (how-to). Maximizing declarativity, will improve quality and productivity, while minimizing software development and customization mistakes and therefore costs;
7. Semantic web awareness, in special respecting its distributed, ever-evolving and incomplete nature (of both ontologies and instances), together with the application and exploitation of proposed ideas and technology.

While not corresponding to “research goals”, the expression “quality vectors” captures and reflects the endeavor towards better ontology mapping solutions. Therefore, these vectors should be maximized during this research work.

4.2 MAFRA Overview

The distributed nature of Semantic Web entails significant degrees of information redundancy, incoherence and constant evolution, thus changing the nature of the ontology mapping problem.

MAFRA provides an approach and conceptual framework that provides a generic view of the overall distributed mapping process. It organizes the requirements resulted from Chapter 2 into a

coherent, useful representation of the ontology mapping process. Those requirements can be divided into two categories:

- Operation requirements:
 - Identification, specification and representation of syntactic, schematic and semantic relations between distinct ontologies;
 - Transformation of information exchanged among intervenients according to the specified syntactic, schematic and semantic relations;
- Complementary operation requirements:
 - Tools to support the negotiation capabilities to reach consensus;
 - Tools to support the maintenance of the syntactic, schematic and semantic relations;
 - Tools that integrate (but minimize) the human-being intervention in the mapping process;

Semantic web awareness, the sixth requirement identified in 2.5.7, is not understood as a foundational requirement, but as an implementation-level issue.

The previous categorization is centered in the objects being manipulated, while a process is conceptually concerned with the types of manipulations upon the objects. Therefore, prior categorization is more useful in this context if refined into:

- Operation requirements:
 - Identification of
 - Specification of
 - Representation of
 - Transformation of data according to

| syntactic, schematic and semantic relations between
 | distinct ontologies;
- Complementary operation requirements:
 - Negotiation capabilities to reach consensus;
 - Maintenance of the syntactic, model and semantic relations;
 - Integration
 - Minimization

| of the human-being intervention in the mapping process.

In addition to prior requirements, two more operational requirements have been identified and included in MAFRA:

- Translation of ontologies (or schemas) and respective data to a common model and nomenclature is necessary because it makes semantics differences between the source and the target ontology more evident [Bernstein & Rahm, 2001; Sheth & Larson, 1990]. This requirement has been identified in section 2.1 as a complete distinct process in the five-level integration architecture. Yet, the MAFRA suggested ontology mapping process does not contradict that perspective, but proposes their close interrelation. In fact, according to [Sheth &

Larson, 1990], the translation process may require not only an automated, rule-driven data model translation, but also subjective, semantic-driven translation;

- Analysis of the resulting transformation instances aims to detect errors or inconsistencies between the target repositories and the resulting transformation instances. In particular, two situations are necessary to address:
 - Detection of invalid target instances, namely those not respecting constraints defined in the target ontology (e.g. “Person must have a positive age”);
 - Detection of duplicate instances, which occurs when a target instance seems to be an already existent target instance. Two situations may then occur: (i) the duplication really exists and (ii) the duplication do not exists.

According to all previous requirements, the ontology mapping process has been systematized and concentrated into the MAFRA diagram presented in Figure 4.1.

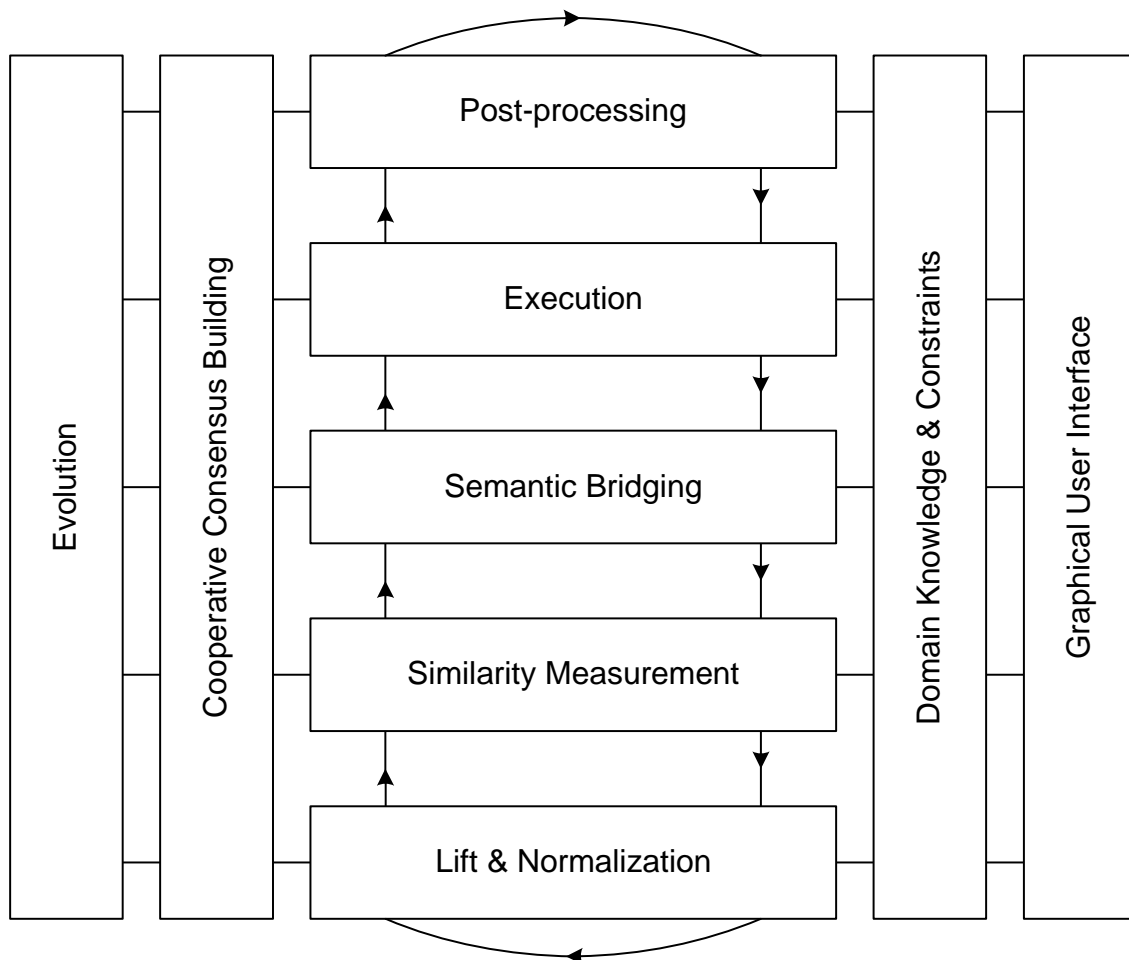


Figure 4.1 – MAFRA – MApping FRamework

The framework follows the categorization previously made, clearly separating the operation requirements from the complementary operation requirements.

MAFRA diagram is therefore organized according to two dimensions:

- The horizontal dimension, is concerned with the operation requirements, representing the ontology mapping process;
- The vertical dimension, is concerned with the complementary operation requirements, representing and providing crucial functionalities along the entire ontology mapping process, even if do not belonging to the core process.

The two next sections describe respectively the horizontal and vertical dimensions. Each section describes the conceptual goals and participation in the ontology mapping process of each module²¹, along with the respective state-of-the-art. The third next section describes the interrelation between modules and the flow of the ontology mapping process.

4.3 Horizontal Dimension of MAFRA

The horizontal dimension of MAFRA respects the process core operations.

4.3.1 Lift & Normalization

This module corresponds to the translation requirement identified previously. Besides the translation process goals, addressed by the Lift part of this module, the Normalization part aims to homogenize ontologies contents in respect to their lexical layers.

4.3.1.1 Lift

The Lift sub-process is responsible for the unification of data model and representation language of the ontologies. Even if some important knowledge is neglected from original representations, this step allows the process to focus on ontologies contents rather than the form [Bernstein & Rahm, 2001; Omelayenko & Fensel, 2001; Stuckenschmidt & Wache, 2000]. This unification respects three types of translation operations (Figure 4.2):

- The translation of the ontologies schema, defined according to a specific data model, into an ontology whose schema respects the common data model;
- The translation of the ontologies instances represented according to the original ontology schema into the new ontology schema;
- The translation of the ontologies and respective instances from the original representation syntax (language) into the common representation language (CRL). The fact an ontology respects the CDM does not mean it is represented in the CRL (e.g. RDFS, OIL, DAML and

²¹ The components of a framework are referred as modules. Moreover, because the framework also represents the ontology mapping process (the process), modules are also referred as sub-processes.

OWL ground to the same data model but they use different syntaxes), and therefore this translation must be considered in the process.

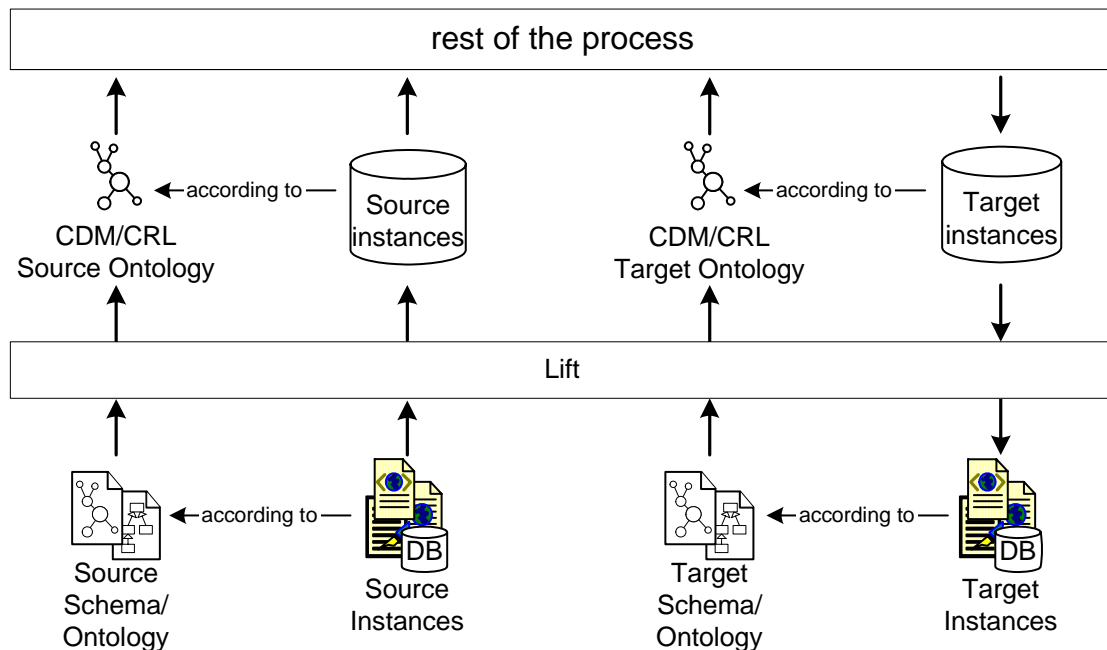


Figure 4.2 - Figurative representation of the input and output of the Lift sub-process

Notice that the resulting target instances are initially represented according to the CDM and CRL target ontology, thus the translation occurs in the opposite direction.

Schema translation is a well studied problem both in the database [Atzeni & Torlone, 1995; Sheth & Larson, 1990] and ontology [Chalupsky, 2000; Fodor *et al.*, 2002; Omelayenko, 2002b; XSLT] research communities.

One of the most common technical approach to operationalize this process in distributed environments is the use of wrappers (responsible for the syntax translation), and mediators (responsible for data model translation) [Wiederhold & Genesereth, 1995].

4.3.1.2 Normalization

The Normalization sub-process is responsible for the lexical normalization of ontologies contents. It aims to get both ontologies the most lexically similar as possible, providing that no semantic commitment occurs.

Example 4.1 – Normalization without semantic commitment

Ontology O1 defines concept “id” and ontology O2 defines concept “identification”. Apparently, no semantic commitment is performed if ontologies are unified to either “id” or “identification”.

Example 4.2 – Normalization with semantic commitment

Ontology O1 defines concept “name” and “identification”. Even both concepts meaning partially overlap, some semantic commitment is necessary to unify them. Thus, no translation should be done in this phase.

This operation is particularly important for the automatic identification of similarities between ontologies entities. The more lexically harmonized ontologies are less noise is introduced into the similarity measuring sub-process.

The most typical normalization operations include, but are not limited to:

- Expansion (or contraction) of:
 - Acronyms (e.g. PC → Personal Computer, H₂O → water, e.g. → example);
 - Abbreviations (e.g. id → identification, ex → example);
- Tokenization (e.g. PersonalComputer → Personal Computer; personal_computer → Personal Computer);
- Letters (de-) capitalization (e.g. PERSON → person, person → Person).

However, the most important factor of lexical heterogeneity is due to the development of ontologies in different natural languages (e.g. Portuguese, English). Despite the fact that some ontology representation languages support multi-language terminology (e.g. RDFS lexical layer extensions provided by KAON [Motik *et al.*, 2003]), it is not common practice developers associate multi-language terminology with ontological entities. In such cases, it is necessary to translate the original terminology into a “common natural language”, which provides the minimal support for automatic similarity measurement.

However, natural language translation comprises many semantic commitments, which further improves ambiguity to the next phases of the ontology mapping process. Similar problems may surge from the other normalization operations too, even if the risk is smaller. As consequence, normalization sub-process must be carefully analyzed and developed much semantically independent as possible, even if this implies its smaller role in the mapping process.

Each of these normalization operations is a complex, independent and orthogonal problem to many domains and research communities [Guarino, 1997a; Kahn & Hovy, 1997; Resnik, 1999; Silva & Rocha, 2002]. These communities deal with these problems for long (e.g. Natural Language Processing), and many methods and tools have been developed, which may be useful to the normalization sub-process. Additionally, specific knowledge bases (e.g. glossaries) upon the ontologies domain of discourse are helpful in supporting the sub-process. These knowledge bases and operational generic tools are found in the “domain-knowledge & constraints” module.

4.3.2 Similarity Measuring

The Similarity Measurement phase aims to discover and measure similarities between source ontology entities and target ontology entities. The task associated with this sub-process is also known as matching [Bernstein & Rahm, 2001; Doan *et al.*, 2002; Madhavan *et al.*, 2001; Milo & Zohar, 1998; Noy & Musen, 2001; Rahm & Bernstein, 2001].

The similarities measures derived in this sub-process will be applied on the Semantic Bridging sub-process by explicitly grouping entities and stating the relation holding them. Similarities are stated by domain expert or automatically discovered by computer-based systems. However, it is commonly accepted that this task is intrinsically and inherently subjective [Guarino, 1994; Madhavan *et al.*, 2001; Noy & Musen, 2000; Sheth & Larson, 1990] and therefore no complete solution can be expected from a completely automatic process.

In the databases scope, yet valid in the context of ontologies, Sheth and Larson [Sheth & Larson, 1990] claim that “one reason why a completely automatic schema integration process (particularly for discovering attribute relationships) is not possible is because it would require that all of the semantics of the schema be completely specified. This is not possible because, among other reasons, (1) the current semantic (or other) data models are unable to capture a real-world state completely, (2) it will be necessary to capture much more information than is typically captured in a schema, and (3) there can be multiple views and interpretations of a real-world state; and the interpretations change with time. Convent [1986] formally argues that integrating relational schemas is undecidable.”

This sub-process aims to measure similarities between ontological entities the more semantically accurate as possible.

Data model elements such as *is_a* (subclass of) and σ (interrelation between classes) are primarily used to derive similarity, specially recurring to graph-based analysis. The *is_a* and the σ relations form graphs that can be analyzed to derive similarity. However, these elements provide insufficient semantics [Bergamaschi *et al.*, 1999; Madhavan *et al.*, 2001], since similar graphs can exist with no or small semantics in common.

Example 4.3 – Graph-based similarity measuring scenario

Consider the scenario of Figure 4.3 where the schemas of two simple ontologies are represented in UML. Because both ontologies are very similar in their graph representation and interpretation, the graph-based similarity would reach 100%.

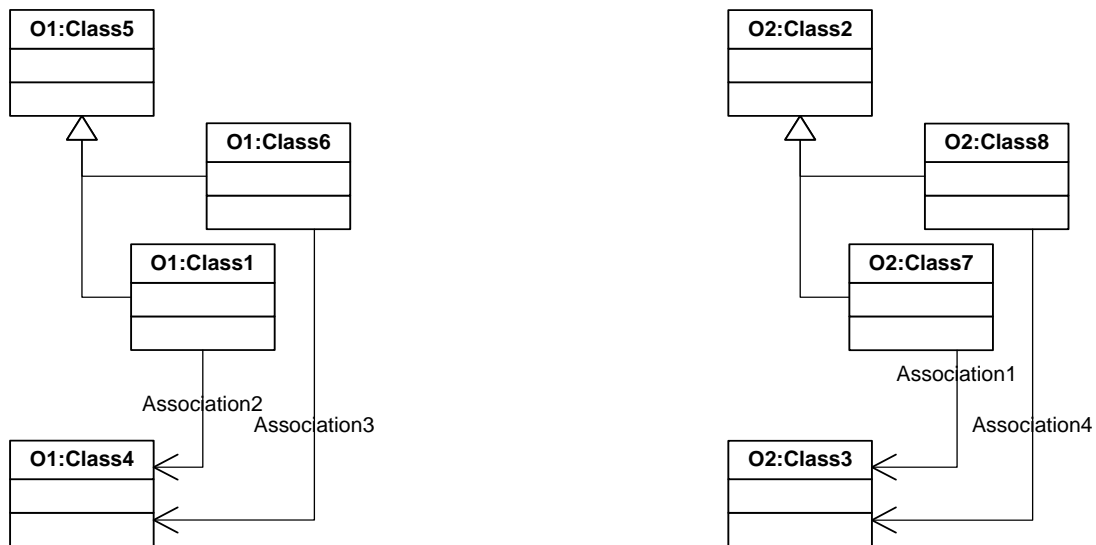


Figure 4.3 – Graph-based similarity measuring scenario

However, it often occurs that terms associated with the ontological entities are very similar, very different or make no sense, allowing critical similarity conclusions. If the previous ontologies scenario is refined in order to include entities meaningful labels, other conclusion would naturally arise (Figure 4.4).

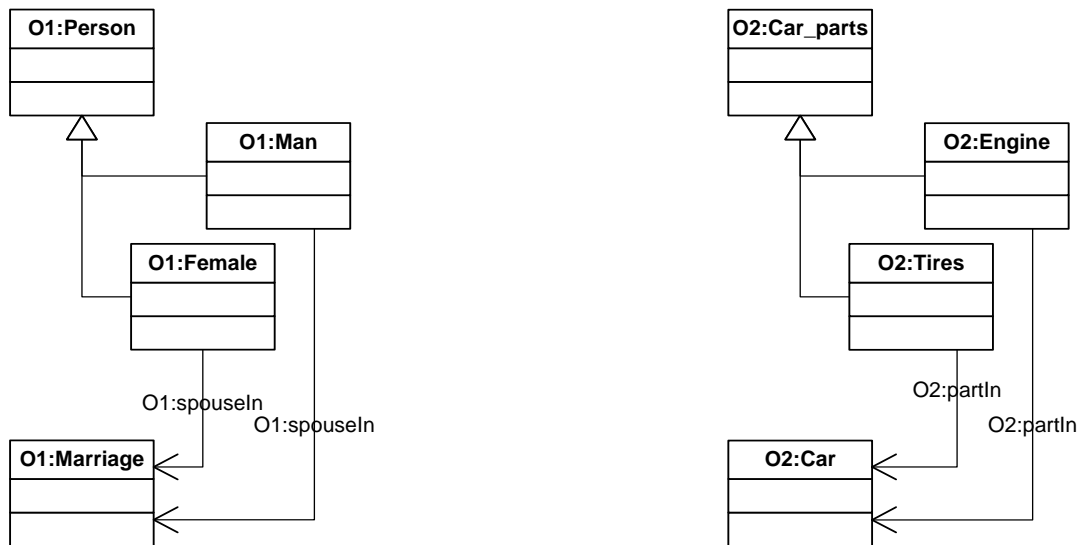


Figure 4.4 – Similar scenario but providing meaningful entities labels

Besides its pertinence, graph-based similarity is indeed insufficient to derive accurate similarities between two ontologies entities.

When available, the lexical layer of ontologies supports extended similarity measuring potentialities. Lexical terminology has the ability to relate ontology entities with real-world object, through independent knowledge bases, such as natural language dictionaries, thesaurus and other lexical structures like WordNet [Miller *et al.*, 1990]. These knowledge bases capture the meaning senses

associated with lexical terms in common real-world contexts and, in some of them, extra relations connect the terms, providing extra semantics to reason upon (e.g. hyponyms, meronyms).

However, the identified knowledge bases capture the semantics in very generic and multiple domains of discourse, constraining the accuracy of the similarity.

Example 4.4 – Meaning senses of “person” according to Merriam-Webster dictionary

When querying Merriam-Webster Online dictionary [Webster] for the English lexical term “person”, seven distinct possibilities are returned:

1. HUMAN, INDIVIDUAL -- sometimes used in combination especially by those who prefer to avoid man in compounds applicable to both sexes <chairperson> <spokesperson>;
2. a character or part in or as if in a play: GUISE;
3. (a): one of the three modes of being in the Trinitarian Godhead as understood by Christians; (b): the unitary personality of Christ that unites the divine and human natures;
4. archaic: bodily appearance; (b): the body of a human being; also: the body and clothing <unlawful search of the person>;
5. the personality of a human being: SELF;
6. one (as a human being, a partnership, or a corporation) that is recognized by law as the subject of rights and duties;
7. reference of a segment of discourse to the speaker, to one spoken to, or to one spoken of as indicated by means of certain pronouns or in many languages by verb inflection.

This kind of answer is ambiguous. On one side, the increased number of possibilities increases the matching chances. On the other side, as consequence of the previous, even poorly semantically related ontologies entities turn out to be somehow related.

Word Sense Disambiguation (WSD) is a very active research area aiming to determine the correct sense of the lexical term in a specific context [Dionísio *et al.*, 2001; Resnik, 1999], which can be of great help in this process. However, most of the WSD approaches deal with word-sense, hand made, classified *corpora*²², which in turn suggests its subjective and limited nature, in both extension and coverage, constraining the quality of the similarity measures [Mitra & Wiederhold, 2001].

Other approaches, such as analysis of attribute types and cardinalities of concepts properties are also used. In [Rahm & Bernstein, 2001] authors suggest a taxonomy for the classification of schema matching approaches, i.e., the approaches applied in deriving matches between schema entities. The

²² “A collection or body of knowledge or evidence; especially: a collection of recorded utterances used as a basis for the descriptive analysis of a language” in [Webster].

taxonomy presented in Figure 4.5 is the adaptation of their taxonomy to the ontology entities similarity measuring.

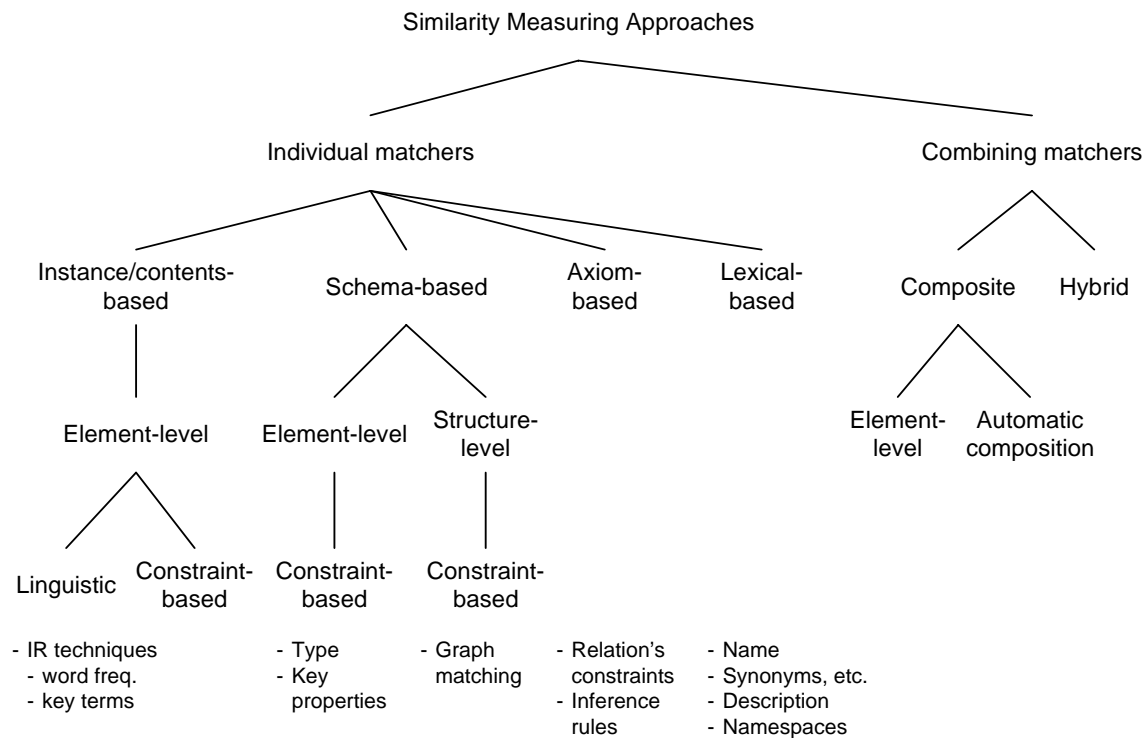


Figure 4.5 – Taxonomy of schema matching approaches

The presented taxonomy differs from the original [Rahm & Bernstein, 2001] in two ways:

- Lexical-based approaches have been clearly distinguished from the schema-based approaches. This is due to the fact that, typically, additional and more complex lexical terminology is available in ontologies than in schemas. Therefore, new lexically-based similarity evaluation approaches should be considered;
- Axiomatic-based approaches are introduced, in order to explicitly exploit the axiomatic layer of ontologies (refer to 3.3.3).

Some recent research work focused in exploiting elements typically absent from schemas but typically present in ontologies, such extra lexical descriptions (especially used in information retrieval) and axioms (which is insipient due to the lack of axiomatic layers in ontologies). However, partially arising from the Semantic Web initiatives, most of the efforts have also focused on the analysis of instances as source of knowledge. Probabilistic [Doan *et al.*, 2002], statistical [Kang & Naughton, 2003], machine learning [Doan *et al.*, 2001] and clustering [Beneventano *et al.*, 2001] are some of the most relevant strategies used to evaluate similarity between ontology instances.

However, no individual approach, nor a combination of any of the individual approaches, is sufficiently good for every situation. In particular, two limitations are noticed:

- Only some are capable to determine n:1 matches [Madhavan *et al.*, 2001; Mitra *et al.*, 1999] and none is capable to determine 1:n matches;
- None is capable to determine or suggest the transformation to occur between the entities.

4.3.3 Semantic Bridging

The Semantic Bridging phase establishes expressions correlating a set of source ontology entities with a set of target ontology entities through a transformation function. These expressions are named “Semantic Bridges”, as they permit to overcome semantic heterogeneity between information repositories. The set of semantic bridges between two ontologies is named Ontology Mapping Document.

This sub-process has three types of inputs:

- The set of similarities calculated in the similarity measurement sub-process, which help determine which source ontology entities are semantically related to which target ontology entities;
- The transformation functions available in the system, which constrain the semantic bridges to establish between ontologies entities;
- Execution and post-processing sub-processes error reports:
 - Execution will report (i) the duplicate transformation of the same instance, and (ii) the insufficiency of instances to transform;
 - Post-processing will report the existence of multiple (different, complementary and/or overlapping) instances for the same real-world object in the target repository.

The semantic bridging sub-process is characterized according to three dimensions: (i) Automation, (ii) Specification methods and (iii) Representation language.

4.3.3.1 Automation

The automation dimension concerns the ability of the system to propose semantic bridges between ontologies entities according to the previous inputs. As mentioned during the early chapters of this thesis, no complete automation of the process is possible and therefore human intervention upon the proposals is envisaged.

If a manual approach is considered, the similarity-measuring phase is not mandatory, once the domain expert implicitly establishes the matches when specifying the semantic bridges. Yet, if automatic support is applied, the domain expert can exploit the similarity measures that

automatically arise, limiting the search space. In an automatic process, the similarities measures resulting from prior phase are essential, especially to:

- Determining which source ontology entities are to be grouped together and bridged to which target ontology entities. It can happen that the similarities resulting from previous phase already group entities into semantically related entities, but this is not mandatory and even not suggested in some situations. In fact, the grouping tasks is conceptually associated with the semantic bridging phase and not with the similarity measurement phase;
- Determining the transformation function to apply between the set of source and target ontology entities in each group of related entities;
- Associating each ontology entity to the correct parameter of the transformation function.

Several projects based on manual approaches have been proposed in recent years [Dou *et al.*, 2002; Madhavan *et al.*, 2002; Park *et al.*, 1998; Stuckenschmidt & Wache, 2000], but no automatic approaches are currently available, except the one developed in the scope of this thesis and presented in 7.3. In case of manual operation, the problem of semantic bridging is limited to the specification method and representation language.

4.3.3.2 Specification methods

This dimension concerns with the characteristics of the methods used to specify the semantic bridges. According to Park and colleagues [Park *et al.*, 1998], there are three type of mappings:

- Implicit mappings are those, which for (at least) one of the intervenient systems is adapted to meet the other(s) intervenient systems information requirements. This type of mapping requires systems to change their perspective (or at least their representation) of the domain of knowledge, which is not always possible or beneficial. In some aspects, implicit ontology mapping and ontology merging process are very similar, particularly because in both cases systems are mutually changed to meet the other system characteristics;
- Procedural mappings, which are defined using transformation code encompassing the logic necessary to transform instances between repositories. This type of mapping focus on the operation necessary to transform the instances (*how-to*);
- Declarative mappings are those that are specified through declarative statements that require interpretation during the Execution sub-process. Declarative mapping focus on the description of *what-to* instead of *how-to*.

In the scope of this thesis only declarative mappings are considered, especially because:

1. Repositories should be kept separated and independent, maintaining their own semantics. This requirement directly collides with the premises of implicit mappings;

2. Procedural mappings are very dependent on the implementation language, which can be a problem when applied in multiple distinct scenarios, as described in Chapter 2. Instead, declarative mappings are independent of both the language and the systems platform, since they rely on an independent interpreter to be executed;
3. Declarative mappings are more explicit and direct, allowing even non-experts to maintain the mapping document. Maintainability of the ontology mapping document is a fundamental requirement identified in 2.5.7;
4. Declarative mappings naturally orient human-being participation to semantic decisions instead of syntactic specifications, which contribute to raise the profile of the human-being contribution, while minimizing his/her participation in low-profile tasks, has requested in 2.5.7;
5. Conversely to the traditional coding interfaces, graphical user interfaces (GUI) ennoble the human-being participation. Because GUIs are intrinsically declaratives approaches, the relation between the (declarative) method and its operationalization through GUI is facilitated. Further descriptions about the GUI subject are found in 4.4.4.

4.3.3.3 Representation language

The languages to represent the ontology mapping document are directly influenced by the representation method chosen. Because only declarative methods are considered, language must conform to this constraint.

As ontology mapping is a very specific problem, generic declarative languages (e.g. Lisp, Prolog) will require new primitives to meet the ontology mapping requirements, especially respecting the meta-specification. This has been the solution adopted in [Dou *et al.*, 2002], where a Lisp-like language named Web-PDDL, a strong typed first-order logic language for web application. Yet, when conveyed through the web, the ontology mapping document (including the bridging axioms) is translated into DAML. In [Stuckenschmidt & Wache, 2000], authors suggest the use of a Prolog-like logic language to specify semantic relations, while in [Stuckenschmidt & Visser, 2000], authors apply OIL axioms (Ontology Inference Layer) [Fensel *et al.*, 2000] into the FaCT reasoner [Horrocks, 1998] [Silva, 2002b].

Another alternative is to define an ontology of semantic bridges, which will serve as an ontology mapping document when instantiated. This is the approach first adopted in Park and colleagues work [Park *et al.*, 1998], to map between knowledge-bases and problem-solving methods (PSMs)²³. RDFT (RDF Transformations) [Omelayenko, 2002b] and SBO (Semantic Bridging Ontology)

²³ Recently, Crubézy and colleagues [87] further adopted the Park and colleagues work to map between knowledge-bases and PSMs described according to the UPML (Unified Problem-solving Method development Language) [Fensel *et al.*, 1999].

[Maedche *et al.*, 2002b; Silva & Rocha, 2003a; Silva & Rocha, 2003d], have been recently proposed in the scope of the Semantic Web.

Furthermore, the language must respond to the ontology mapping system requirements, namely concerning their capabilities to overcome different types of semantic heterogeneities (refer to 5.2). Finally, as required in 2.5.7, the representation language must be sensitive to the Semantic Web environment.

Further details and comparison between SBO, RDFT and other approaches are provided in 5.3.

4.3.3.4 Outlook of Semantic Bridging

The Semantic Bridging sub-process represents one of the most demanding tasks in the overall process, in which the human-being is highly required and not substitutable. Multiple human-oriented approaches currently exist, but no automatic approach provides more than simple equivalence relations between pair of entities derived from the similarities resulted from previous phase.

By instantiating an ontology of semantic bridges, RDFT, SBO and the approach by Park and colleagues propose a new semantic bridging approach, in which the semantic bridging ontology serves both as reasoning mechanism and as ontology mapping representation language.

4.3.4 Execution

The Execution phase²⁴ transforms instances from the source ontology into target ontology instances by evaluating the ontology mapping document defined in the semantic bridging phase. The ontology mapping system makes sense only if this phase is completely automatic. Four distinct dimensions characterize this sub-process: (i) classification process, (ii) transformation process (iii) entity-driving execution and (iv) operation mode. These dimensions are further analyzed in next sections.

4.3.4.1 Classification process

This dimension refers to the method applied in categorizing instances from the source ontology into the target ontology.

In many aspects and scenarios, the representation language of semantic bridges and the execution process are directly and closely dependent.

²⁴ It is also referred as Transformation phase.

Traditional declarative languages are normally associated with existent logical reasoners (theorem provers), such as FaCT (in case of OIL), OntoEngine (in case of WebPDDL) and Protégé (in case of KIF) or TRIPLE [Sintek & Decker, 2002].

Comment 4.1

Logical reasoners are normally used to check ontologies for consistency and for computing subclass relations not explicitly contained in the ontology [Horrocks, 1998]. Subclasses are specified according to constraints defined upon subclasses²⁵, which are in turn explicitly or implicitly defined in the ontology²⁶.

Instances of implicit subclasses (axiom-defined subclasses) may be explicitly stated as instances of implicit classes, or inferred from their values (intentional view). Refer to Example 3.3.

This approach has the advantage of allowing both data-driven and query-driven transformation, which represents an immediate solution to the source and target instance-oriented transformation (described in next section).

While the use of general logical reasoners is an immediate and practical solution in systems already familiar with the technology, it has normally performance limitations. Depending on the ontology and knowledge-bases representation languages, equivalent solutions are available through query languages²⁷ such as SQL (relational databases), OQL (object-oriented databases) [Cattel *et al.*, 2000], XQuery (for XML documents) [XQuery], or RQL (its implementation in Sesame [Broekstra *et al.*, 2002; Karvounarakis *et al.*, 2002]) (for RDFS schemas). Because of the Semantic Web awareness, SQL and OQL are not directly relevant in this context. Besides related to the WWW, XQuery is also inappropriate in the scope of this work because it does not abstract enough from the tree-structure of the XML document in order to address the ontology model [Broekstra *et al.*, 2002].

As functional languages, previous RDFS query languages provide the result of a query over an RDF document as an RDF document again, which can be further and incrementally queried. Unlike logical reasoners, query languages perform satisfactorily and scale well. Currently, no ontology mapping projects using ontology query languages are known.

²⁵ This type of constraint has been referred in 4.3.2 as inference rules.

²⁶ The specification cycle ends when the entity (e.g. class) is defined upon an explicitly defined entity (e.g. class).

²⁷ Query languages rely on query engines which in turn are implemented either by logical or imperative languages.

4.3.4.2 Transformation process

The transformation process is related with the operation executed upon source ontology instances so they become target ontology instances. Concatenation, split, copy and arithmetic functions are examples of transformation functions.

In the context of the Semantic Web, the XSL Transformation (XSLT) [XSLT] arises naturally as strong hypothesis to apply in the process. XSLT is a declarative language used to transform a source XML document into other XML document, which is a very similar task to that at hand. However, as been referred for the XML Query language, the XSLT does not abstract enough from the tree-structure of the document. During early phases of this thesis, doubts remained about the XSLT applicability to the ontology mapping process, but it is now clear that the abstraction provided by the ontology is of fundamental importance in minimizing the human intervention and in semantically enriching the ontology mapping document and process. As consequence, XSLT or schema-oriented transformation language should not be conceptually used.

Through their associated language (e.g. Prolog, Lisp, TRIPLE), logical reasoners also provide some mechanisms to transform properties instances. Whereas logic reasoners provide extensive and extensible transformation mechanisms, they are not fully suited for functional transformation. Similar assumptions are valid for query languages too.

One important observation from the scenarios described in Chapter 2 it is the unpredictable number and types of transformations required. In fact, heterogeneity between source and target ontologies is so diverse that it is always possible to find a mapping situation not covered for a certain set of functions. The same observation is valid for the semantic bridging phase, where the transformation function is one of the elements to be defined into semantic bridges (see 5.2).

4.3.4.3 Entity-driving execution

This dimension characterizes the execution according to the type of the entity driving the process. It analyzes the transformation process according to the type and number of entities involved in the execution process.

Five types of entities driving the execution process are envisaged:

1. Semantic bridge. Because every semantic bridge relates a source ontology concept to a target ontology concept, each instance of the source ontology concept is transformed according to the semantic bridge into the target ontology concept instance;
2. Source ontology concept. Because each source ontology concept can be semantically bridged to multiple target ontology concepts, each source instance can give rise to multiple target instances;

3. Target ontology concepts. Because each target ontology concept can be semantically bridged from multiple source ontology concepts, multiple semantic bridges will be executed over multiple source ontology concepts, giving rise to multiple target instances;
4. Source ontology instance. Because a source instance belongs to (at least²⁸) one source concept, and because one source concept might be bridged to multiple target concept, each source instance can give rise to multiple target instances;
5. Target ontology instances. This type of execution is also referred as a query because some of the required characteristics of the target instance are specified. When all characteristics of the target instance are specified, the process aims to determine if exists a source instance that satisfies the specified requirements. Because a target instance belongs to (at least) one target concept, and because one target concept might be bridged from multiple source concepts, each query might take multiple source instances and give rise to multiple target instances.

Table 4.1 resumes previous analysis:

Table 4.1 – Type and number of entity involved in the execution process

Involved entities →	Semantic Bridges	Source Concepts	Target Concept	Source Instance	Target Instance
↓ Driving entity					
Semantic Bridge	1	1	1	Many	Many
Source Concept	Many	1	Many	Many	Many
Target Concept	Many	Many	1	Many	Many
Source Instance	Many	1(Many)	Many	1	Many
Target Instance (Query)	Many	Many	1(Many)	Many	1

The OntoMerge project supports both source and target instance-oriented execution [Dou *et al.*, 2003] through the logical inference system OntoEngine. In fact, all logical inference-based systems, as referred in 4.3.4.1, inherently support these two types of transformations.

4.3.4.4 Operation mode

This dimension corresponds to the moment the mapping execution is performed. Two distinct modes of operation are envisaged:

- Offline (or static) method corresponds to the execution of the mapping upon a source repository before the need for the transformed instances. Typically, this method is further characterized by:
 - Scheduled or seldom executions;
 - High computational-load concentrated on time.

²⁸ By axiomatic inference, a specific concept instance can be instance of another (implicit) concept.

As consequence of these characteristics, repositories are often unsynchronized. Because synchronization is not a fundamental issue in data warehousing, it is often applied in such application scenarios;

- Online (or dynamic) method corresponds to the execution of the mapping as the target intervenient requires the instances from the source intervenient. This method is further characterized by:
 - Continuous execution;
 - Short-time executions;

This is the proper method when synchronization between repositories is required or when momentary, unscheduled executions are necessary. Federated databases, E-Business and information retrieval are typical application scenarios for this method.

The other scenarios analyzed in Chapter 2 have different requirements depending on the specific application.

While the off-line method poses no problem to the system, the on-line approach requires better communication and synchronization mechanisms, which are not intrinsically supported by logical inference systems. In [Park *et al.*, 1998] authors claim the pertinence of both offline and online methods, but no further implementation details are referred, even in consequent descriptions [Crubézy & Musen, 2003].

4.3.4.5 Outlook of Execution

While the semantic bridging phase is intrinsically subjective, the execution phase is intrinsically objective and automatic. Most of the research projects in this area are based on logical inference systems. This execution approach has several advantages, such as the intrinsically ability to execute both instance and query-oriented transformations. However, this approach has some drawbacks too, especially concerning with performance and computational loads. Besides, no approaches based on query languages are currently available, preventing further analysis about the feasibility of the approach.

4.3.5 Post-processing

Post-processing phase aims to check and increase the quality of the target instances resulting from the execution phase. Verification is done according to three elements:

- The target ontology, especially because the semantic bridges specification might not completely respect the target ontology.

Example 4.5 – Inconsistencies between semantic bridges and the target ontology

Consider a scenario where the cardinality of O1:Individual.name is not stated, while O2:Person.has_name is constrained to 1. Imagine that a semantic bridge is stated in a way that the values of O1:Individual.name are copied to O2:Person.has_name, which specifies no constrain about the cardinality of the transformation. Thus, a KB such as:

O1: Individual(i_1), name(i_1 , "Nuno"), name(i_1 , "Silva")

will be transformed into:

O2: Person(i_1), has_name(i_1 , "Nuno"), has_name(i_1 , "Silva")

which is clearly an ontological mistake.

In fact, semantic bridges might be under-specified or inconsistent with the target ontology, leading to invalid ontology instances;

- The target knowledge base. One of the most common errors occurring at instance-level respects object-identity. Object-identity concerns the recognition that in the target knowledge-base:
 - Two (or more) distinct instances represent the same (real-world) object. This situation occurs because of either:
 - The same instance exists in both the source and target knowledge base;
 - The semantic bridges are under-specified, allowing the creation of false distinct objects;
 - Two (or more) similar entities represent two different (real-world) objects. This situation occurs because:
 - These two (or more) false similar instances already existed in the source knowledge base;
 - The semantic bridges are under-specified, allowing the creation of false similar objects;
- The semantic bridges. It can happen that semantic bridges precede errors that are detected but not solved during the execution phase. These errors might not evidence ontological or object-identity mistakes but conversely some instances might not be transformed as required.

No references are currently known about ontology mapping systems with support for this process. However, independent research exists in this field. The conceptual work of Guarino and Welty [Guarino & Welty, 2000] on metamodeling, in which identity, unity, rigidity, and dependence metamodeling primitives are studied, provide a good conceptual background for this problem. Complementary, extensive work on Verification and Validation (V&V) of knowledge and database systems [Coenen *et al.*, 1999] is a good starting point for a pragmatic solution to these problems.

Yet, some of the problems previously identified are much dependent on the semantic bridging and execution phases. Therefore, significant research has to be done both in addressing these problems and in combining previous research into the ontology mapping process.

4.4 Vertical Dimension of MAFRA

Four complementary modules have been identified in the vertical dimension of MAFRA, which are described in the following sections.

4.4.1 Evolution

The evolution module aims to manage the ontology mapping document according to external factors. At least three types of external factors affect the ontology mapping document:

- Changes in the source and target ontologies;
- Changes in the domain or application requirements;
- Changes in the mapping mechanism, namely update in the transformation capabilities.

This module will focus in providing supporting mechanisms to the core process phases, in two distinct tasks:

- Maintenance of the ontology mapping document, according to external requirements;
- Versioning of the ontology mapping document, according to changes in the external requirements.

While research on evolution of ontology mapping is missing, extensive research exist concerning evolution [Stojanovic *et al.*, 2002a] and versioning [Klein *et al.*, 2002] of ontologies in the context of the Semantic Web. In particular, Stojanovic and colleagues suggest the concept of an evolution strategy capable to drive the user-requirements in the process, while allowing the customization and control of the strategy. Klein and colleagues focus on the management of ontology versioning in the Semantic Web, especially respecting the conceptual and transformation relations maintained between different ontology versions.

While these research works are a valuable starting point for this module, considerable research is still expected. Ontology mapping evolution is very different from ontology evolution, suggesting that the ontology evolution strategy proposed will not fit the ontology mapping specificities. A similar assumption is possible respecting versioning.

4.4.2 Cooperative Consensus Building

This module aims to support and promote consensus between two (or more) interoperability intervenients, in two important tasks related to the ontology mapping process:

- The specification and maintenance of the ontology mapping document;
- The version of the ontology mapping document to use in certain interoperation.

An interesting interpretation of this module, which is in fact an envisaged application of consensus building tasks, is the improvement of the quality of the mapping specification. Exploiting and capturing the know-how of distinct, eventually more competent, third-party entities into the mapping specification, the quality of the mapping specification will potentially increase. Under this perspective, cooperative consensus building is not an end *per se* but a mean to achieve better ontology mapping.

Research in the area of meaning negotiation is much related with this subject, but its status suggests the need for intensive research in this area. In [Bailin & Truszkowski, 2001] authors propose a simple framework providing support for three fundamental tasks: interpretation, clarification of terms and evolution of ontology. While the framework is extensible it is considered clearly insufficient or even not suited to the ontology mapping problem. In fact, the envisaged ontology mapping problems are much more complex than those addressed by Bailin and Truszkowski. Similar comments should be done concerning the Sarini and Simone works [Sarini & Simone, 2002]. Conversely, a more conceptual approach is proposed by van Elst and Abecker [van Elst & Abecker, 2002], which better reflects the ontology mapping process. Anchor-Prompt, suggested by Noy and Musen [Noy & Musen, 2001] is a human-driven supporting tool for ontology matching, which might be a starting point towards the automation of the process.

Yet, the final opinion about the state-of-the-art in this subject is that most of work corresponds to very simple approaches, sometimes even naïve, requiring further intensive research efforts.

4.4.3 Domain Constraints and Background Knowledge

The quality of similarity measurement and the automatic semantic bridging may be dramatically improved by introducing two complementary elements in the process:

- Background knowledge, respecting common sense descriptions of the world, including:
 - Dictionaries of lexical terms, which textually describe multiple senses of the lexical entity;
 - Dictionaries of translations between different idioms;
 - Thesaurus, which provides synonyms of a lexical entity;
 - Glossaries of acronyms and abbreviations;
 - Other lexical tools such as WordNet [Miller *et al.*, 1990], which has the ability to correlate lexical entities through a large variety of relations (e.g. synonymy/antonymy, hyponymy/hypernymy²⁹, meronymy/holonymy³⁰);

²⁹ Hyponym/hypernym relation corresponds to the ontological *is_a* relation. For example, Woman is a hyponym of Person because Woman *is_a* Person. Conversely, Person is an hypernym of Woman. These relations are transitive.

- Domain constraints, respecting specific perspectives of the ontology domain (e.g. Genomic, Sports, Medicine, Electronics). Domain constraints include but are not limited to:
 - Glossaries of domain acronyms and abbreviations;
 - Domain thesauri;
 - Domain glossaries³¹;
 - Standards specifications, which provide relations between components of products or services, comparable to the ontological relations *is_a* and *has_a*;

Notice that the meaning, components or properties of an entity vary according to knowledge source, which is a consequence of the distinct views of the universe. This feature would eventually lead to ambiguity problems.

Furthermore, exploiting previously systematized human-defined knowledge bases, this module provides support for automatic operation, thus reducing the human-being intervention in the process.

As observed in [Rahm & Bernstein, 2001], much of the projects concerned with similarity measurement apply background knowledge, but it is not so common the use of domain constraints. Even if similarity measurement and semantic bridging phases are those that most clearly profit from this module, no mapping phase should neglect the potential usefulness of these knowledge sources.

4.4.4 Graphical User Interface

Mapping is a difficult and time-consuming process, which is not less difficult than building an ontology itself. Ontology mapping process require deep understand of both ontology conceptualizations and their semantic similarities. Special difficulties arise during the specification phase of the process, thus requiring human intervention. Moreover, browsing structure of both ontologies, definition and customization of semantic relations, all require extensive manipulation support.

A Graphical user interface (GUI) is therefore a fundamental module in an ontology mapping system, allowing and promoting better ontology mappings, while minimizing human efforts.

³⁰ Meronym/holonym relation corresponds to the ontological relation *has_a/is_part_of*. For example, wheel is a meronym of car because wheel *is_part_of* car, and *vice-versa*. These are transitive relations.

³¹ An extensive list of glossaries in several languages and domain can be found in: <http://www.jump.net/~fdietz/glossary.htm>

Currently, most of the ontology mapping systems represent the ontology either as text (e.g. KIF, RDFS) or as a tree-like structure. However, typical ontologies are not trees but graphs. Tree-like representation hides substantial part of the semantics of the ontologies, which are far from being appellative and truly supportive. This is the case of the work of Crubézy and colleagues [Crubézy *et al.*, 2003] in the scope of Protégé. In the scope of this thesis however, a fully functional, net-based GUI has been developed, integrating several phases of the ontology mapping process (refer to 8.1.4).

4.5 Ontology mapping process flow

The ontology mapping process envisaged in the scope of MAFRA grounds on the idea that an ontology mapping document, as any product or system, exhibits a life cycle. The ontology mapping process, as the responsible for the ontology mapping document, adopts a cyclic perspective too, so it can fit the ontology mapping document manipulation requirements. Furthermore, the ontology mapping process exhibits the following characteristics:

- Incremental, because the document is improved in every phase of the process;
- Interactive, because the human-being is the ultimate responsibility for the flow of the process;
- Continuous, since the ontology mapping document can always be improved, especially if the automatic methods are applied.

The outcome of each phase serves either as the input of the next phase (except in case of the post-processing phase) or as feedback to previous phases (except in case of the Lift & Normalization phase). Besides the cyclic nature of the process, some phases are not mandatory. In fact, all but the semantic bridging and the execution phases are optional.

4.6 Summary

The MAFRA – MApping FRAmework has been described in this chapter. The three main goals subjacent to its specification have been extensively addressed, providing a valuable tool for analysis and comparison of further approaches and projects. While the state-of-the-art research has been enumerated and briefly described, its focus was on the different perspectives of each sub-process, even if in several cases no current research is known or available. On the other hand, in case multiple and very distinct research approaches exist, the description focused on the most recent and close related approaches, according to the identified requirements.

Complementary analysis and comparison of projects and approaches are further presented during the rest of this thesis, but then under a more specific and defined context.

Chapter 5

SEMANTIC BRIDGING

This chapter describes the semantic bridging phase of the MAFRA – MApping FRAmework. The work described in this chapter has been previously published in [Maedche *et al.*, 2002b; Maedche *et al.*, 2002a; Silva & Rocha, 2002; Silva & Rocha, 2003a; Silva & Rocha, 2003d; Silva & Rocha, 2004b].

The chapter is divided into seven main sections. In the first section the semantic bridging phase is formalized in respect to the ontology mapping process. The second section makes a short analysis of semantic heterogeneity occurring between two information repositories according to related research. Grounding on this analysis, third section proceeds with a fine grained analysis of semantic relations necessary to overcome such semantic heterogeneity. According to this analysis, a systematization of the semantic relations dimension is derived, which is further applied in defining envisaged support this work will provide for the semantic bridging phase of the process. Based on the envisaged support, a rather focused analysis of related research work is provided in fourth section, which provides inspiration for the rest of the work by revealing their advantages and

limitations. The fifth section describes of the Semantic Bridging Ontology, one of the core parts of this thesis. In sixth section, it is presented an annotated example of the application of the Semantic Bridging Ontology to a mapping scenario. Seventh section draws a comparison between the envisaged, the effective support and that provided by related works.

5.1 Ontology mapping: two-phases process

5.1.1 Informal definition

Despite ontology mapping characterization made in Chapter 4, ontology mapping is primarily the process whereby semantic relations are defined at ontological level between source ontology entities and target ontology entities; and then further applied at instance level, transforming source ontology instances into target ontology instances. Figure 5.1 exposes this perspective.

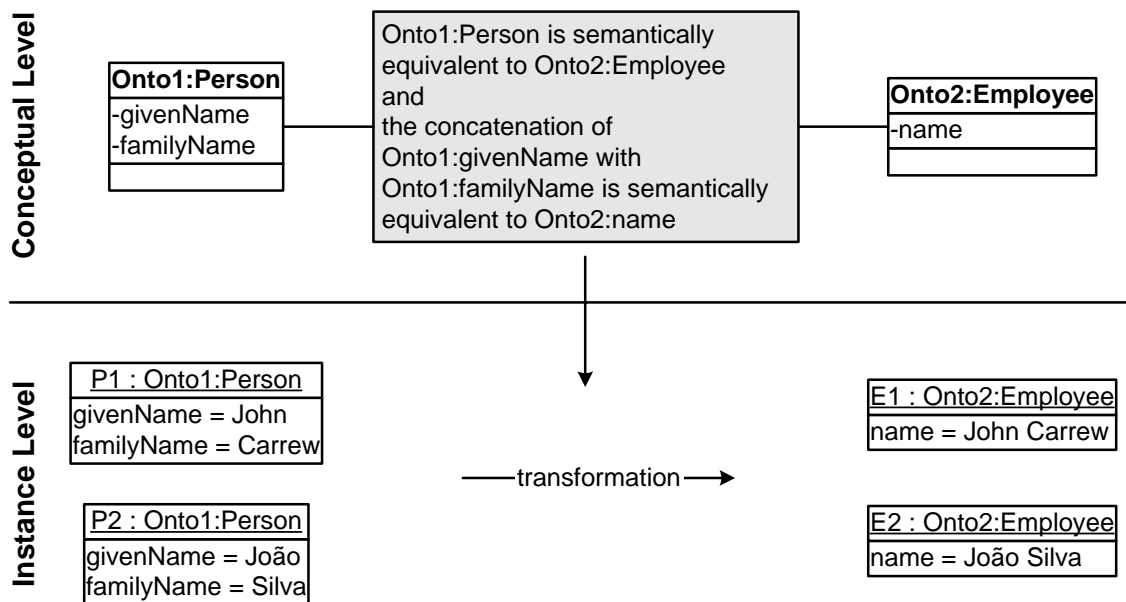


Figure 5.1 – Informal representation of ontology mapping

Ontology mapping does not intend to unify ontologies and their data, but to transform ontology instances according to the semantic relations (mapping relations) defined at conceptual level. Repositories are therefore kept autonomous and heterogeneous, maintaining their complete semantics and contents unchanged.

5.1.2 Formal definition

Formally, ontology mapping is also described as a two-phase process [Silva & Rocha, 2003d]. The first phase, named (semantic bridging) specification phase, is formally defined as a relation between source and target ontology entities:

$$\mathcal{M} \subseteq \mathcal{E}^s \times \mathcal{E}^t$$

\mathcal{M} is not defined as a function, but as a relation, because the definition aims to encompass the fact that:

- \mathcal{M} rarely maps all ontology entities from one model into the other. It would be therefore referred as a partial function;
- \mathcal{M} often relates the same source ontology entity more than once, i.e. the same domain element relates to different co-domain elements.

In a mathematical context “mapping” expression would be misused in the sense that the mapping term is used as synonym of function. However, the expression “ontology mapping” is widely accepted in this context [Kalfoglou & Schorlemmer, 2003], and will be used for the rest of the work.

\mathcal{M} is the ontology mapping specification, or simply ontology mapping, containing the necessary and sufficient information required to transform, during execution phase, source ontology instances into target ontology instances.

The goal of this phase is therefore the specification and representation of \mathcal{M} according to the semantic of both (source and target) ontologies, complemented by domain expertise as referred below. Ontology mapping specification is a meta-level process in the sense that the manipulated objects represent the domains of instance-level elements. The ontology mapping document will be further described in section 5.4.7.

The second phase, named (semantic bridging) execution phase is formally defined as a relation between source knowledge base and target knowledge base, parameterized according to the ontology mapping document developed in the meta-level phase:

$$\mathbf{T}(\mathcal{M}) \subseteq \mathcal{I}^s \times \mathcal{I}^t$$

Ontology mapping execution phase occurs at instance level, transforming source ontology instances into target ontology instances, according to the specified ontology mapping (\mathcal{M}).

5.2 Semantic heterogeneity

As referred in previous chapters, three types of heterogeneities may arise between ontologies:

- Syntactic heterogeneity, which refers to the use of different representation nomenclatures, notations or syntaxes (e.g. OWL, UML, XOL, natural language);

- Model heterogeneity, which refers to the fact that different data models are used to describe the structure and organization of the information (e.g. OO, Relational, frame-based models);
- Semantic heterogeneity, which is due to different ontological commitment respecting distinct perceptions of universe³² [Sowa, 1999]. It occurs when ontologies differently represent elements of the world, or when “disagreement occurs about the meaning, interpretation or intended use of the ontology elements” [Sheth & Larson, 1990].

Whereas important conflicts arise from syntactic and model heterogeneity, this section specially focuses on the identification and analysis of semantic heterogeneities arising in the context of semantic bridging.

Semantic heterogeneity has been studied for a long time [Fodor *et al.*, 2002; Hammer & Medjahed, 1993; Sheth & Larson, 1990; Stuckenschmidt & Wache, 2000; Visser *et al.*, 1997] but no agreement exists on what semantic heterogeneity formally is, nor it is possible to enumerate all its facets.

Goh [Goh, 1997] suggests the analysis of semantic heterogeneities according to three types:

- Confounding conflicts, occur when identical ontological representation correspond to distinct domain (real) elements;
- Scaling conflicts occur when distinct units are used to measure (characterize) the concept;
- Naming conflicts occur when distinct terms are used to represent identical domain (real) objects.

Hammer and colleagues [Hammer & Medjahed, 1993], aiming to resolve semantic heterogeneity in ontology merging scenarios, enumerate the following list of semantic relations:

- Identical relation, occurring when both concepts are “exactly the same”;
- Equivalent relation occurs when both concepts are conceptually identical, but the internal composition is differently specified. For example, one concept is composed by an attribute which is described in two parts in the other concept;
- Incompatible concepts relation occurs when no semantic similarity exists between both concepts;
- Compatible relation occurs when concepts are “neither incompatible nor equivalent”. Accordingly, not all instances of the source concept can be transformed into the target concept. This relation further covers two other sub-types:
 - Specialization/generalization relation, which occurs when one of the concepts is more generic/specific than the other;
 - Positive “association”, which occurs when, in some context, concepts are interchangeably

³² “the fact we live in the same world does not mean we all agree with it or see it the same way” [cited by Patrick Hayes in International Semantic Web Conference 2003, Sanibel Island (FL), USA]

used.

Visser and colleagues [Visser *et al.*, 1997] analyze the ontological mismatches and their influence in interoperability. The analysis systematizes and identifies a set of ontological mismatches according to the schematic (conceptual mismatches) and the axiomatic (explanation mismatches) dimensions of the ontology. For each type of mismatch, the analysis determines if a conceptual solution exists or not. The conceptual solutions subsequently proposed correspond to a set of possible ontological commitments that can be further suggested to (and applied by) the domain expert. However, these can hardly be transformed into heuristics and further applied in (semi-automatic) semantic bridging system. In fact, these solutions are case-based, which has severe limitations especially in ontology mapping where the types of semantic relations are unpredictable. Yet, the conclusions are very concise in distinguishing between:

- Manageable mismatches, which are those that can be solved;
- Hard mismatches, are those where only a difficult or unfeasible solution exists;
- Unknown mismatches, which mean the solution “depend on the case at hand”.

While those analyses are important, they do not provide much information on the characteristics and components of semantic relations occurring between ontologies entities, which represent a fundamental input when developing a (semi-automatic) semantic bridging system.

Characterization of semantic relations

Unlike previous referred works, the analysis of semantic relations made in the scope of this work aims to identify and characterize their components. This approach allows better specification of requirements and possibilities of the envisaged system.

Five components (dimensions) have been identified in the scope of this thesis [Maedche *et al.*, 2002b; Silva & Rocha, 2003a]:

1. Entity type dimension, which reflects the type of ontological entities being related;
2. Transformation dimension, which relates to with the function to transform instances;
3. Cardinality dimension reflects the number of ontology entities being related;
4. Constraint dimension, which respects the constraints that hold during the execution phase;
5. Structural dimension, which reflects the relations between semantic relations.

These dimensions are further analyzed in the following sections.

5.2.1 Entity type dimension

The entity type dimension considers the type of the ontology entities being related. This dimension depends on the types available through the ontology representation language, and on the considered notion of ontology. In fact, the representation language may provide a superset of the

types considered in the notion of ontology. For instance, the representation language can provide the lexical entity type while it is not considered in the ontology. According to Section 3.3, the notion of ontology in the scope of this thesis considers the types: (i) concepts (or classes), (ii) properties (either relations or attributes), (iii) lexicons and (iv) axioms.

However, it is important to keep in mind the goal of the ontology mapping process: transform instances of source ontology entities from the source knowledge base into ontology instances of the target knowledge base. Lexicons and axioms are ontological entities that are not instantiated in the knowledge base. Hence, concepts and properties are the only types of entities to transform.

According to the entity types the following semantic relations can exist:

- Concept to Concept, e.g. O1:Person bridges to O2:Employee;
- Concept to Property, e.g. O1:Person.address.Address bridges to O2:Employee.address;
- Property to Property, e.g. O1:Person.name bridges to O2:Employee.name;
- Property to Concept, e.g. O1:Person.supervisor bridges to O2:Employee.managedBy.Employee;
- Entity to Instance, e.g. O1:Professor bridges to O2:Job, i.e. the class O1:Professor will give raise to an instance of O2:Job.

5.2.2 Transformation dimension

This dimension characterizes the transformation occurring between source and target instances. This dimension is of fundamental importance in the characterization of the semantic relations, and has influence in other dimensions. The following analysis focuses on three characteristics:

- Function;
- Directionality;
- Completeness.

5.2.2.1 Function

Source instances are transformed into target instances according to the specified function. This is eventually the most characterizing element of the semantic bridge, since it reveals much about the semantic bridge semantics.

Example 5.1 – Several transformation functions

1. Copy, creates a target instance with the same contents of the source instance;
2. Concatenation, concatenates multiple source instances into a single target instance;
3. Split, separates a single source instance into multiple target instances;
4. Table-based translation, transforms a certain source instance according to a mapping table;

5. Arithmetic function, transforms a set of source instances into a single target instance according to an arithmetic function (e.g. addition, multiplication);
6. Default value function, would create target property instances with predefined (default) values.

Notice, however, that the transformation function dimension is not, nor can be, fully characterized, once the functions to apply depend ultimately on the ontology mapping scenario at hand. Therefore it is unpredictable the set of functions the mapping system must support.

5.2.2.2 Directionality

Directionality reflects the capability of the semantic bridge to transform instances in both directions. This characteristic applies both to the function and to the semantic bridge.

The semantic bridge is intrinsically bidirectional if the function is bijective³³. Conversely, if the function is injective, surjective or neither surjective nor injective, the semantic bridge is intrinsically unidirectional. In case bidirectionality is required and initially applied function is not bijective, the inclusion in the semantic bridge of extra elements is necessary. In particular, if the inverse relation exists (see 5.2.2.3) the specification of the inverse function and the correlation of ontologies entities and functions arguments is a fundamental input.

Directionality is normally not related to the applied function, but to the ontology mapping scenario, since the function is applied in respect to the semantic relation holding between ontology entities. If the semantic relation is bijective, a bijective function may be applied. However, bijective relations (functions) are not so common, and even in case an inverse transformation is possible/exists, it is not straightforward how to apply or to customize it.

Example 5.2 –Inverse functions: concatenation vs. split

Imagine that a semantic bridge with concatenation function is stated between O1:Person.givenName, O1:Person.surname and O2:Individual.name. Consider that a blank space is set between given name and surname values.

For the instances givenName(i₁, “Gabriel”), surname(i₁, “García Márquez”), the transformation would result in name(i₃, “Gabriel García Márquez”)

Because the concatenation function is not bijective, it is necessary to find an inverse function. The split function is the obvious inverse function, but the blank character mentioned earlier, is not sufficient to play the role of split character). In fact, these transformation scenarios are non-deterministic except if a concatenation/split character is used that undoubtedly determines the split location.

³³ A bijective function is a function that is simultaneously injective (on-to-one) and surjective (onto), which means that every domain element is associated with exactly one codomain element and each codomain element is associated with exactly one domain element.

5.2.2.3 Completeness

Completeness is the characteristic of the transformation that considers the loss of information in the transformation process. A transformation is complete if no information is lost (lossless) and incomplete when information is lost (lossy). Two reasons contribute for the completeness of the semantic relation:

- The characteristics of the ontologies, especially different granularity and generality (see 3.1);
- The transformation function. If the transformation (function) element is ill-specified or unavailable the semantic relation is incomplete.

If the transformation is incomplete, no inverse transformation is available/possible without domain expertise (see 5.2.3) or without loss of information.

Example 5.3 – Transformation completeness: loss of information

Consider the ontology O1 represents the account information according to O1:Account.credit and O1:Account.debt, and ontology O2 represents only O2:Account.balance. The granularity of information is much finer in ontology O1. Accordingly, a semantic bridge from O1 to O2 is possible but impossible from O2 to O1.

Notice that this characteristic can also be applied to the ontology mapping document as a whole. In addition to generality and granularity, incomplete ontology mapping documents are also due to representation of different domains. Because this characteristic depends, above all, on the ontologies and semantic relations holding between them, it is out of control of the representation language of semantic relations.

Despite the characterization of semantic relations according to completeness, the same characteristic is used in categorizing the execution process with a similar meaning. In fact, in the scope of the execution system, completeness concerns with the capability to transform as much as possible source instances into target instances. This characteristic is further addressed in Chapter 6.

5.2.3 Cardinality dimension

This dimension represents the number of ontologies entities at both side of the semantic bridge, i.e., the number of entities whose instances are transformed (source ontology entities) and the number of entities instantiated (target ontology entities).

Cardinality is represented in the form of x:y, where x represents the number of source entities, and y the number of target entities, ranging from 0:1 to m:n

- 0:1 cardinality, which typically corresponds to the specification of a default value in the target instance. This cardinality can be generalized to 0:n, but it conceptually corresponds to n times 0:1 semantic relations (e.g. for each instance of O1:Woman, specify O1:Person.gender=="feminine");

- n:0 semantic relations make no sense if unidirectional transformation is supported. However, if bidirectionality is supported, this cardinality is translated into 0:n in the opposite direction which makes sense, though;
- 1:1 semantic bridges are those in which one source entity instance is necessary and sufficient to create one target ontology instance. (e.g. O1:Person.age bridges to O2:Employee.age). This cardinality is a specific case of both 1:n and n:1 semantic relations;
- 1:n cardinality semantic bridge, are those which one source entity instances gives raise to multiple ontology entities instance (e.g. O1:Person.name bridges to O2:Employee.givenName and O2:Employee.surname);
- n:1 cardinality semantic bridges are those in which one instance of multiple source entities is necessary to create one target entity instance (e.g. O2:Employee.givenName and O2:Employee.surname bridges to O1:Person.name);
- Semantic relations with cardinality m:n are uncommon and in most cases they can be (easily) decomposed into two semantic bridges of cardinality m:1 and 1:n.

5.2.4 Constraint dimension

The constraint dimension permits to control the execution of a semantic bridge. The constraint manipulation process includes:

- Specification of conditions at semantic bridging phase;
- Instantiation and evaluation of conditions at execution time.

Due to distinct ontology modeling decisions, namely concerning with generality and granularity, the execution process is often dependent not only on the ontologies entities being mapped but in third party elements. Three situations might occur:

- The ontology entity and the instance are sufficient to determine all elements of the transformation (e.g. O1:Person bridges to O2:Person);
- Other ontological entities and respective instances are necessary (e.g. O1:Person bridges to O2:Address foreach O1:Person.address);
- Extra ontology data is necessary (e.g. O1:Person bridges to O2:Man if O1:Person.gender=="masculine").

One interesting application of the constraint dimension is the specification of the number of target instances to be created by each semantic relation. Through this dimension, it would be possible to create one target instance from each source entity (not source instance), which would in fact correspond to the Entity to Instance semantic relation, as referred in 5.2.1.

5.2.5 Structural dimension

This dimension reflects the way elementary semantic relations may be combined into complex semantic bridges. There are three main reasons to combine semantic bridges:

- The object-oriented modeling approach provided by the `is_a` ontological property, permits inheritance of properties between concepts. Semantic bridges defined between two concepts would benefit from the semantic bridges specified between their super-classes;
- The property-centric modeling approach provided by the domain and range ontological properties, allows a property to be part of multiple classes. Such properties would eventually be transformed by the same semantic bridge independently of the domain concept it is defined in;
- Control the flow of execution of semantic bridges according to constraints.

5.2.6 Summary of characterization

The characteristics of semantic relations between two ontologies have been identified, analyzed and systematized according to five distinct characteristics, referred as dimensions. This multi-dimensional characterization of semantic relations (summarized in Table 5.1) provides an easily perceptible and manageable framework upon which it is possible to categorize semantic relations.

Table 5.1 – Summary of the multi-dimensional characterization of semantic relations

Dimensions		Characteristics
1. Entity type		Concept to Concept
		Concept to Property
		Property to Concept
		Property to Property
		Entity to Instance
2. Transformation	2.1 Function	Basic functions required
		Combination of functions
		Integration of new functions
	2.2 Directionality	Unidirectional
		Manually bidirectional
		Automatically Bidirectional
3. Cardinality		0:1
		0:n
		1:1
		1:n
		n:0
		n:1
		m:n

4. Constraint	Not constrained
	Ontology entities-based
	Non-ontology entities-based
5. Structural support required	Object-oriented
	Property-centric
	Flow execution control

Distinguishing between and characterizing these dimensions instead of categorizing the semantic relations as a whole, allows a more versatile and applicable categorization process. For example, one can describe a specific semantic bridge as a “unidirectional, concatenation of 3:1 Property to Property, not constrained semantic bridge”.

Categorization of semantic relations is a very important step toward an ontology mapping solution once it provides the elements to limit the problem and address it accordingly. In that respect, previous characterization and analysis suggests important directions towards modeling, specification and development of the semantic bridge representation language. In particular, this analysis distinguishes the components in the semantic bridging phase and their role in the execution phase.

Despite the fact the prior framework provides a mechanism for categorizing semantic relations, it is necessary to determine which types will be allowed and supported by the semantic bridge representation language in particular, and by the rest of the system in general. Table 5.2 describes the envisaged support:

Table 5.2 – Envisaged support according to the characteristics of semantic relations

Dimensions		Characteristics	Envisaged support
1. Entity type		Concept to Concept	Yes
		Concept to Property	Yes
		Property to Concept	Yes
		Property to Property	Yes
		Entity to Instance	Limited
2. Transformation	2.1 Function	Set of basic functions	Yes
		Combination of functions	Yes
		Integration of new functions	Yes
	2.2 Directionality	Unidirectional	Yes
		Manually bidirectional	Limited
		Automatically Bidirectional	Limited

3. Cardinality	0:1	Yes
	0:n	Combining n 0:1 bridges
	1:1	Yes
	1:n	Yes
	n:0	Limited
	n:1	Yes
	m:n	Combining m:1 and 1:n bridges
4. Constraint	Not constrained	Yes
	Ontology entities-based	Yes
	Non-ontology entities-based	Yes
5. Structural support	Object-oriented	Yes
	Property-centric	Yes
	Flow execution control	Yes

This characterization is helpful in systematizing the context and problems addressed in the applicability vector identified in 4.1. In fact, this quality vector is directly and ultimately dependent on the categories of semantic relations supported by the ontology mapping system. Because the semantic relations are now fine grained characterized, it becomes possible to state an upper limit to this quality vector. So, in the context of this work, the upper limit of the applicability quality vector is the support for the entire set of categories of semantic relations just described. However, because the transformation dimension is ill-specified, it is necessary to propose solutions even if not formal.

5.3 State of the art

Once the characterization of semantic relations and their support in this system is specified, it is possible to analyze existent solutions and determine their suitability according to requirements and quality vectors. While applicability is the only quality vector whose upper limit has been (at least partially) specified, pair-wise comparison between projects is possible and advisable. The following research works have been analyzed and compared:

1. Protégé approach;
2. Stuckenschmidt and colleagues approach;
3. RDFT approach;
4. OntoMerge approach.

In the next sections the pros and cons of every one of these research approaches according to the specified requirements will be analysed. Notice that both Protégé and Stuckenschmidt and colleagues works describe research efforts running for several years before the work developed in this thesis. Yet, effective results of Protégé have been mostly presented in the period of this thesis. The RDFT and OntoMerge approaches have been developed and firstly presented in scientific events where this work has been presented too.

5.3.1 Protégé

The work described in this section runs in the scope of Protégé since 1994 [Gennari *et al.*, 1994], and is a very relevant work in the ontology mapping research field. Despite the fact that Protégé has a very wide knowledge based systems application, the work described here corresponds to the research on the mapping between knowledge bases (KB) and problem solving methods (PSM). The efforts aim to develop methods and tools to reuse KBs and PSMs, by the transformation of the knowledge bases contents in respect to the PSMs requirements and *vice-versa*.

The work described by Gennari and colleagues [Gennari *et al.*, 1994] is probably the first attempt to systematize and describe ontology mapping relations through an ontology. This ontology serves not only as a description of the KB to PSM mapping domain of knowledge, but also as ontology mapping representation language, when instantiated in specific mapping scenarios. The types of relations defined in this ontology are:

- Renaming relations, which are able to copy the values of the source instances into the target instances;
- Filtering relations apply filters (constraints) and/or transformations (functions) to create the target instances from source instances;
- Class relations, which are able to fill up target instances according to information captured from source classes (entities) instead of their instances.

While the renaming relation type is far insufficient, filtering relations are able to cope with complex heterogeneity problems. However, as referred in [Gennari *et al.*, 1994], specification of filtering relations might not be easy. Class relations correspond, according to the previously defined terminology, to the Entity to Instance semantic relation.

An important driving concern is the simplicity of the types of semantic relations, about which authors claim that if complex semantic relations are necessary, then the user or domain expert is suggested to programmatically (procedurally) implement the mapping.

Later, the approach [Park *et al.*, 1998] was expanded as a result of the feedback received in the application of the first mapping ontology. A valuable set of desiderata and mapping dimensions is presented, representing the background for the work described in previous section (5.2). According to the achieved desiderata, the type of semantic relations described has been redefined to encompass the following types of semantic bridges (referred as mapping):

- Instance mapping, which corresponds to the Concept to Concept and (possibly to the) Property to Concept types of semantic relations. Despite the target instance creation, instance mappings are also used to embrace the mappings responsible for the properties mappings (described next);

- Slot mapping, corresponds to the Property to Property and (possibly to the) Concept to Property types of semantic relations, and are therefore responsible for the creation of target properties from source entities. Slot mappings are further specialized into:
 - Renaming mapping, corresponds to copy instances of source properties to target properties;
 - Constant mapping, which correspond to the 0:1 cardinality types of semantic relations;
 - Lexical mapping, which basically corresponds to the Concatenation type of semantic relations and therefore corresponds also to n:1 semantic relations;
 - Regular-expression mapping corresponds to the application of regular expression-based functions, providing higher transformation possibilities than those of the lexical mapping. The cardinality of this type of mapping ranges from 1:1 to n:m;
 - Numerical-expression mapping is the arithmetic version of the regular-expression mapping;
 - Functional mapping allows “arbitrarily complex transformations” of properties, since it permits “user-supplied” functions to be associated with the mapping.

According to the type of slot mapping defined in, the instance mapping is further specialized into:

- Renaming mapping, in which all slot mappings are of renaming type;
- Direct mapping, in which renaming and constant slot mappings are allowed;
- Lexical mapping, in which renaming, constant and lexical slot mappings are allowed;
- Transformation mapping in which all types of slot mapping are allowed.

According to the authors, this specialization of instance mapping promotes expressiveness and clarity, by characterizing *a priori* the type of slot mappings allowed and (eventually) used.

No published improvements have been released from 1996 until late 2003, when Crubézy and colleagues reviewed the work [Crubézy & Musen, 2003]. Still, not many details were presented then, and apparently the major outcome of this update is the current implementation that exploits the Semantic Web technologies (namely the RDFS-based ontology representation language) and its correlation with the Unified Problem-solving Method development Language [Fensel *et al.*, 1999] in the scope of the Semantic Web. Yet, this is a very valuable reference work in this research domain.

5.3.2 Stuckenschmidt and colleagues

Stuckenschmidt and colleagues [Stuckenschmidt & Visser, 2000; Stuckenschmidt & Wache, 2000] research work concerns with the combination of “context transformation” and “integration rules” for the integration of Geographic Information Systems information sources. Context transformation corresponds to the transformation of data respecting the specificities of a context, into data respecting the semantics of another context. Contexts are conceptualizations of the domain of knowledge, which corresponds to the notion of ontology as introduced in 3.3. Once

contextualized, i.e. once the source data is semantically compatible with target context; it is structurally integrated through integration rules.

Both context transformation rules and integration rules are represented by two other types of rules: combination and replacement rules. In fact, the approach distinguishes between the representation level (combination and replacement rules) and the application level (context transformation and integration rules). While this separation might sound beneficial, it turns out to be difficult to perceive the approach.

Rules are based on the notion of Template, understood as a generic structure for describing information, much like a triple, in some ontology representation languages. A template is a “tuple” in the form of:

$$T := (name, context, type, value) @ source$$

where:

- *name* is the name of the information being described;
- *context* is the metadata describing the information;
- *type* is the data type of value, which can be a primitive type (e.g. string, number, set), or complex;
- *value* is the placeholder for the instances of the information being represented. In case the complex type is stated, this value corresponds to a nested template. Nested templates corresponds, in ontology terminology, to relations between concepts;
- *source* refers to the information source the template belongs to.

Due to the capability to represent nested templates, this generic data model is capable of representing rather complex information such as those represented by relational data model or ontologies.

Rules adopt a logic-like structure composed by a head and a body. In particular, combination rules have the following form:

$$H \leftarrow B_1, \dots, B_n, \phi_1, \dots, \phi_m$$

where:

- H, B_1, \dots, B_n are templates;
- ϕ_1, \dots, ϕ_m are expressions constraining the execution of the relation.

Instead, replacement rules have the following form:

$$H \leftarrow \bar{B} \& B_1, \dots, B_n, \phi_1, \dots, \phi_m$$

where:

- $H, \bar{B}, B_1, \dots, B_n$ are templates;
- ϕ_1, \dots, ϕ_m are expressions constraining the execution of the relation.
- \bar{B} is the template that will replace the head of the rule. Together with a distinct interpretation of the meaning of the rule, this form allows replacement rules to be nested and streamed.

Every element in the template can be represented by a variable, which provides the mechanism to interrelate the various templates, including propagating data from the body to the head.

Example 5.4 – Stuckenschmidt and colleagues semantic relations

Consider the ontology mapping scenario represented in Figure 5.2 where O1 ontology is to be mapped to O2 ontology.

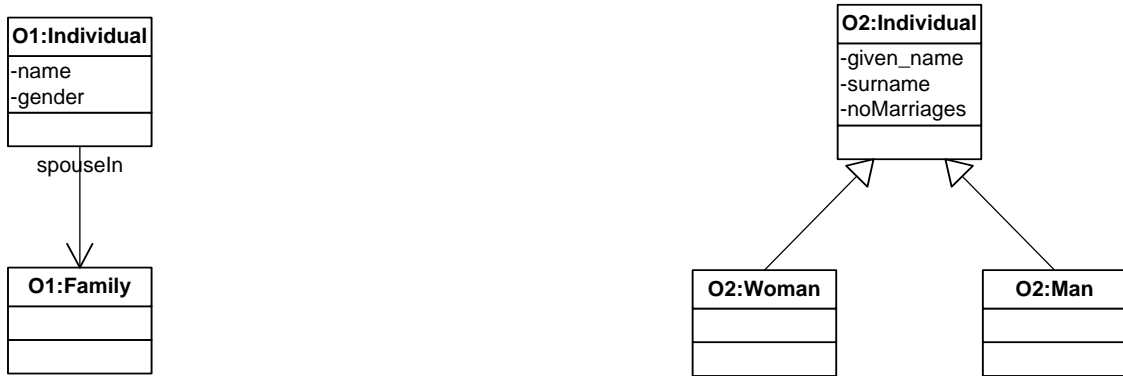


Figure 5.2 - UML representation of two ontologies

O1:Individual is semantically related to O2:Individual, O2:Woman and O2:Man, and O2:Individual.given_name and O2:Individual.surname are filled in with the result of the splitting string of O1:Individual.name by the first white space.

The following replacement rules correspond to both context transformation and integration rules, which do not provides the advocated separation between context transformation and integration:

```

<man,_,complex,
  { given_name → <given_name,_,string,?Given_name>@O2
    surname → <surname,_,string,?Surname>@O2 }
}&@O2
←
<individual,_,complex,
  { name → <name,_,string,?Name>@O1
    gender → <gender,_,string,"masculine">@O1 }
}&@O1 &
<?Given_name,?Surname>=split(?Name,"").
    
```

```
<woman,_,complex,
  {given_name → <given_name,_,string,?Given_name>@O2
    surname → <surname,_,string,?Surname>@O2}
  }@O2
←
<individual,_,complex,
  {name → <name,_,string,?Name>@O1
    gender → <gender,_,string,"feminine">@O1}
  }@O1 &
  <?Given_name,?Surname>=split(?Name,"").
```

The described approach provides no support for object-oriented modeling structure, such as the example above denoted and would require, but instead the same slot mapping (split) has been defined twice, for each target concept.

Nesting and streaming context transformation rules supports, in some extent, the property-centric modeling feature of the ontology representation language, but the process and the relations between rules becomes rather complicated and poorly automatable.

In [Stuckenschmidt & Visser, 2000] authors assume some limitations of this approach, and propose another strategy based on the automatic re-classification of concepts through the standard logic-based reasoners, such as the FaCT reasoner [Horrocks, 1998]. The proposed approach is based on the specification of necessary and sufficient conditions for class (concept) membership, *a la* description logics. From one side, necessary conditions allow the inference of non-explicitly source concept instances, while sufficient conditions allow the inference of target class membership based on properties of source instances. Conditions are represented in PROLOG-like rules, which are then directly forwarded into the FaCT reasoner.

The re-classification partially supports the object-oriented modeling of ontologies in the mapping representation, but because both approaches (rule-based and re-classification) are not integrated, the overall solution, in general, and the representation language in particular, are rather limited.

5.3.3 RDFT

RDFT [Omelayenko, 2002b] has been developed in the same period as the approach proposed and described in 5.4, and has been therefore strongly considered for representation language of semantic relations in the scope of this work. Yet, its expressiveness and transformation capabilities are clearly insufficient. In fact, RDFT has been developed with the B2B catalogue integration application domain in mind, neglecting some important features such as Property to Concept semantic relations. According to Omelayenko [Omelayenko, 2002b], such relations give rise to misunderstandings, and should therefore be treated by procedural-programming based solutions.

However, one of the most important limitations of RDFT is its capability to describe only rather simple regular-expression transformation of attributes.

Despite these important limitations, RDFT has some non-neglectable features. One very important feature is the capability to represent not only relations at conceptual level but also at syntactic level, thus supporting the so called normalization sub-phase of the ontology mapping process (4.3.1.2). In particular RDFT provides constructs to translate from DTD's and XSD's into RDFS models and *vice-versa*. These capabilities are more functional than conceptual though, since in [Omelayenko & Fensel, 2001] authors suggest a two-layer approach (as suggested in 4.2), and not only one as the capabilities of RDFT would suggest.

5.3.4 OntoMerge

In the scope of OntoMerge [Dou *et al.*, 2003; Dou *et al.*, 2002] authors argue that ontology mapping is better understood and advantageous if thought in terms of ontology merging. Ontology merging consists in defining a new ontology (the merged ontology) by the union of both source and target ontologies entities and bridging axioms (semantic bridges in current terminology) representing the semantic relations between source and target ontologies entities. The merged ontology corresponds to whose entities are new representations of the original ontologies. Thus, the merged ontology is itself a fully fledged ontology that can be further merged with other ontologies.

The Web-PDDL language, a Lisp-like, strongly typed first order logic language is used to represent ontologies and bridging axioms. Despite some extra constructs have been added to the Web-PDDL language to support ontology merging especially in the scope of Semantic Web, it is automatically processable by the private but generic OntoEngine reasoner, responsible for the “ontology translation”.

The simplest bridging axiom corresponds in current terminology to the Concept to Concept semantic relation. For every pair of semantically related ontology concepts, a new concept is defined in the merged ontology and two bridging axioms are specified, bridging each ontology concept with the new merged concept.

Example 5.5 – OntoMerge Concept to Concept semantic relations

Consider the example of Figure 5.2, in which O1:Individual is semantically related to O2:Individual. To represent this semantic relation in OntoMerge, one new (merged) concept and two axioms are necessary:

(T->O1:Individual Individual)

(T->O2:Individual Individual)

The $T \rightarrow$ meta-predicate is the type-translation (Concept to Concept) construct,

Despite this type of expressions, all other bridging axioms are first order logic predicates using universal and existential quantifiers, conditions and assertions upon ontologies entities instances. The exception of the notation is the use of namespaces to keep track of the entities origin, i.e. source, target or merged ontology.

Example 5.6 – OntoMerge conditional Concept to Concept semantic relations

The bridging axioms specified above do not correspond to the intended ontology mapping. In order to semantically relate O1:Individual to either O2:Man or O2:Woman, as referred in example of Figure 5.2, more “substantial” axioms are required than those presented above:

```
(T-> @O2:Male Male)
(forall (x - Object)
  (if (is Male x)
    (and (= x (@skolem:mIndividual x) - @O1:Individual)
      (@O1:sex (@skolem:mIndividual x) - @O1:Individual "M"))))
(forall (x - Individual)
  (iff (sex x "M")
    (is Male x)))
(T-> @O2:Female Female)
(forall (x - Object)
  (if (is Female x)
    (and (= x (@skolem:fIndividual x) - @O1:Individual)
      (@O1:sex (@skolem:fIndividual x) - @O1:Individual "F"))))
(forall (x - Individual)
  (iff (sex x "F")
    (is Female x)))
```

Notice the @skolem control responsible for the creation of objects in the target repository that do not exist in source repository. This construct corresponds in current terminology to the Property to Concept semantic relation. Despite this method grounds on well known and established technology i.e. theorem provers, it is feeble expressive and leads to even poorly expressive representation of semantic relations. One of the consequences of this construct is the need to create interrelations between objects at the time they are created, because the identity of the object created (e.g. (@skolem:fIndividual x) is lost outside the bridging axiom it is specified in³⁴.

Other bridging axioms respect the merging of predicates (properties) of the objects (concept instances), which corresponds to the Property to Property semantic relations.

³⁴ This situation is not addressed in presented examples.

Example 5.7 – OntoMerge Property to Property semantic relations

Using again the ontology mapping example of Figure 5.2, the next bridging axioms semantically relates O1:Individual.name with O2:Individual.given_name and O2:Individual.surname:

```
(forall (p - Individual n - @xsd:string)
  (iff (@O1:name p n)
    (name p n)))
(forall (p - Individual n1 n2 - @xsd:string)
  (if (and (@O2:given_name p n1)
    (@O2:surname p n2))
    (name p (+ n1 n2))))
(forall (p - Individual n - @xsd:string)
  (if (name p n)
    (and (@O2:given_name p (@skolem:pName n1) - @xsd:string)
    (@O2:surname p (@skolem:pSurname n2) - @xsd:string)
    (= n (- n1 n2)))))
```

Authors argue that the presented approach permits automatic translation in both directions, but as argued in 5.2.2.2 only in very special cases that would be possible. For instance, notice that in previous example it has been necessary to specify bridging axioms for both concatenation (+) and splitting (-).

Yet, probably the most important limitation of OntoMerge is the completeness of the execution. While this characteristic relates to the execution system and not the representation language, both are tightly dependent and justify addressing the question here. In fact, OntoMerge is based in the OntoEngine inference engine, developed by the authors to overcome type-constraint unification, but just like most theorem-provers it does not guaranty completeness. To overcome this problem, a very simple (even naïf) heuristic rule is used, based on the size of the left and right size of the bridging axiom:

$$W_1 \times \text{size of conclusion} - W_2 \times \text{size of premise}$$

Despite the advantages arising from the fact a new ontology arises from the process, as described by the authors, they ultimately depend on several factors, such as (i) the size of overlaps of merging ontologies and (ii) how the size of the merging ontology change as they are merged. Considering the scale, heterogeneity and dynamicity of the web, these solutions tend to be disregarded in favour of other less centralised and immediate approaches.

5.3.5 Summary

At early stages of this thesis, several works have been analyzed. The Protégé approach and the Stuckenschmidt and colleagues approaches were the most paradigmatic. MOMIS [Beneventano *et*

al., 2001; Bergamaschi *et al.*, 1999] in the database integration field and Clio [Miller *et al.*, 2000] in application of views for information integration have been deeply analyzed too, but their associated representation languages and application scenarios do not corresponds to the identified requirements.

Later, during the development phase of the solution, both RDFT and OntoMerge have come to public knowledge. While OntoMerge is definitely a very good approach in ontology mapping, it is based on private operational components (OntoEngine) constituting a unique solution. A similar situation is observed with MOMIS (the integration system), ODB-tools (the DL and inference system) and the ODL_r³ (representation language). Instead, RDFT is far the less capable of the approaches, but its simplicity and Semantic Web awareness has been a good source of inspiration and comparison.

Table 5.3 compares the observed characteristics of the described approaches with the quality vectors identified in 4.1. Notice that not only characteristics of the representation language are considered in this table, but the overall approach capabilities and limitations.

Table 5.3 – Support of described projects to defined requirements

Requirements	Protégé	Stuckenschmidt et al.	RDFT	Onto Merge	Required support
1. Applicability	(refer to Table 5.4)				
2. Semantic Expressivity	+	+	-	+	+
3. Automation	None	None	None	None	+
4. Modularization	+	-	-	+	+
5. Reutilization	-	+	-	+	+
6. Declarativity	+	+	+	+	+
7. Semantic web-awareness	> 2003	-	+	+	+

One of the most important revelations of this table is the fact that none of the approaches provide automation of the specification of semantic relations. It is tempting to argue that no relation exists between the automation of the mapping process and the representation of semantic relations.

Table 5.4 summarizes the capabilities of the representation languages adopted or developed in the scope of each of these mapping approaches. Some of the characteristics are unknown and are therefore referred as “Unknown”.

Table 5.4 – Support of described projects in respect to the semantic relations characteristics

Dimensions	Characteristics	Protégé	Stuckenschmidt et al.	RDFT	OntoMerge	Required support	
1. Entity type	Concept to Concept	Yes	Yes	Yes	Yes	Yes	
	Concept to Property	Yes	Yes	Yes	Yes	Yes	
	Property to Concept	Yes	Yes	No	Yes	Yes	
	Property to Property	Yes	Yes	Yes	Yes	Yes	
	Entity to Instance	Not anymore	No	No	No	Limited	
	2.1 Function	Set of basic functions	Yes	Yes	Limited	Limited	Yes
		Combination of functions	Yes	Yes	No	Yes	Yes
Integration of new functions		Yes	Yes	No	Yes	Yes	
2.2 Directionality	Unidirectional	Yes	Yes	Yes	Yes	Yes	
	Manually bidirectional	No	Yes	No	Yes	Limited	
	Automatically Bidirectional	No	Limited	No	Limited	Limited	
3. Cardinality	0:1	Unknown	No	No	No	Yes	
	0:n	Unknown	No	No	No	n 0:1 bridges	
	1:1	Yes	Yes	Yes	Yes	Yes	
	1:n	Yes	Yes	Yes	Yes	Yes	
	n:0	Unknown	No	No	No	Limited	
	n:1	Yes	Yes	Yes	Yes	Yes	
	m:n	by combination of 1:n and m:1					
4. Constraint	Not constrained	Yes	Yes	Yes	Yes	Yes	
	Ontological entities-based	Yes	Yes	Limited	Yes	Yes	
	Non-ontological entities-based	Yes	Yes	No	Yes	Yes	
5. Structural support	Object-oriented	No	Limited	No	Yes	Yes	
	Property-centric	Yes	Limited	Yes	Yes	Yes	
	Flow execution control	No	No	No	No	Yes	

According to the previous comparison, one might argue that the much features the language has, the automatic support it provides in comparison with the system possibilities. Empirically however, it is suggested the semantic bridging process focus on the relations allowed and defined in the system. In fact, representation languages should be able to represent all but only the semantic relations the underlying system can process. Manual mapping experiences support this hypothesis by confirming that users tend to relate entities according to a limited set of well known types of transformation functions (e.g. copy, concatenation, split or table translation), instead of a large and unlimited set of possibilities. According to such limited set, the user tries to associate source and target ontologies entities with one of the transformation functions.

Therefore, instead of analyzing the representation language according to its larger capabilities, representation language should be analyzed by its limitations and constraints. Furthermore, representation language should be perfectly declared and represent all but only the system capabilities.

Addressing the problem under this perspective, the representation language represents the limits to the semantic relations, and, as a consequence, the searching space is constrained.

5.4 Semantic Bridging Ontology

The Semantic Bridging Ontology (SBO) describes the ontology mapping domain of knowledge. Exploiting the conceptual and practical characteristics of the ontology (refer to 3.3), SBO has been developed concerning two purposes:

- Capture and describe the knowledge associated with the ontology mapping subject as perceived in this context. The analysis of semantic relations (5.2) and the quality indicators (4.1) just presented, drive its specification;
- Serve as a representation artifact for semantic relations in specific scenarios. An instantiation of the SBO is an ontology mapping document.

SBO has been totally developed in the scope of this thesis, and it has been initially presented in the UML static structure notation in [Maedche *et al.*, 2002b], where all combinations of the entity type and the cardinality dimensions had a correspondent semantic bridge class in the structure. Because the characterization of the ontology mapping domain of knowledge was particularly studied, it considerably diverge from subsequent versions. Indeed, later in [Maedche *et al.*, 2002a], a more pragmatic approach has been derived toward the representation and execution of semantic relations. While current status of SBO resembles to the one presented in [Maedche *et al.*, 2002a], a formal definition of SBO has been introduced [Silva & Rocha, 2004b], including integrity and validation axioms not defined in early publications. These new elements contribute to the formal

comprehension of SBO by heterogeneous and divergent communities, while improving its usefulness to wider range of problems, including the automation feature of the system.

In the next sections the Semantic Bridging Ontology will be described, namely the entities, their inter-relations and constraints. A top-down description and analysis is followed, starting from the core concepts of semantic bridges to the more implementation dependent entities. While the following description is mostly based on logical notation, providing a formal representation, it is complemented with UML notation, which provides an easily perceivable and immediate representation of the subject.

5.4.1 SBO Overview

SBO is an ontology of the ontology mapping domain of knowledge. It specifies, classifies and describes the types of ontology mapping relations, inter relates them and provides other modeling constructs necessary to express ontology mapping documents. Since the ontologies entities are the objects to be mapped, SBO is in fact a meta-ontology.

SBO is fundamentally composed by two distinct but complementary concepts:

- SemanticBridge, which represents the semantic relation between a set of source ontology entities and a set of target ontology entities. It encompasses all information necessary to its correct and univocal interpretation at execution time;
- Service, which represents the transformation capabilities present in the ontology mapping system.

Because SemanticBridge concept does not contain by itself transformation capabilities, neither is able to define the process required to transform source ontologies entities into target ontology entities, it is necessary to associate to each SemanticBridge one specific Service responsible for those roles (Figure 5.3).

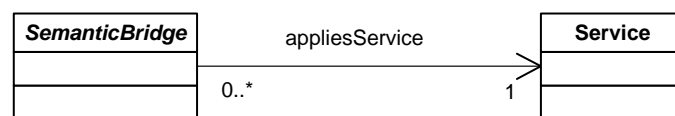


Figure 5.3 - UML representation of SemanticBridge and Service conceptual relation

The semantics and signature of each Service is defined and described according to their source and target Arguments and respective characteristics (e.g. the EntityType). The ArgumentValue class correlates the ontologies entities (i.e. instances of Entity) with the Service Arguments (Figure 5.4). Notice that Entity class does not concern only to ontologies entities, but also with entities composed by the ruled association of ontologies entities (composed entities). In turn, EntityType class corresponds to both the ontology entities and the composed entities (see 5.4.4 through 5.4.6).

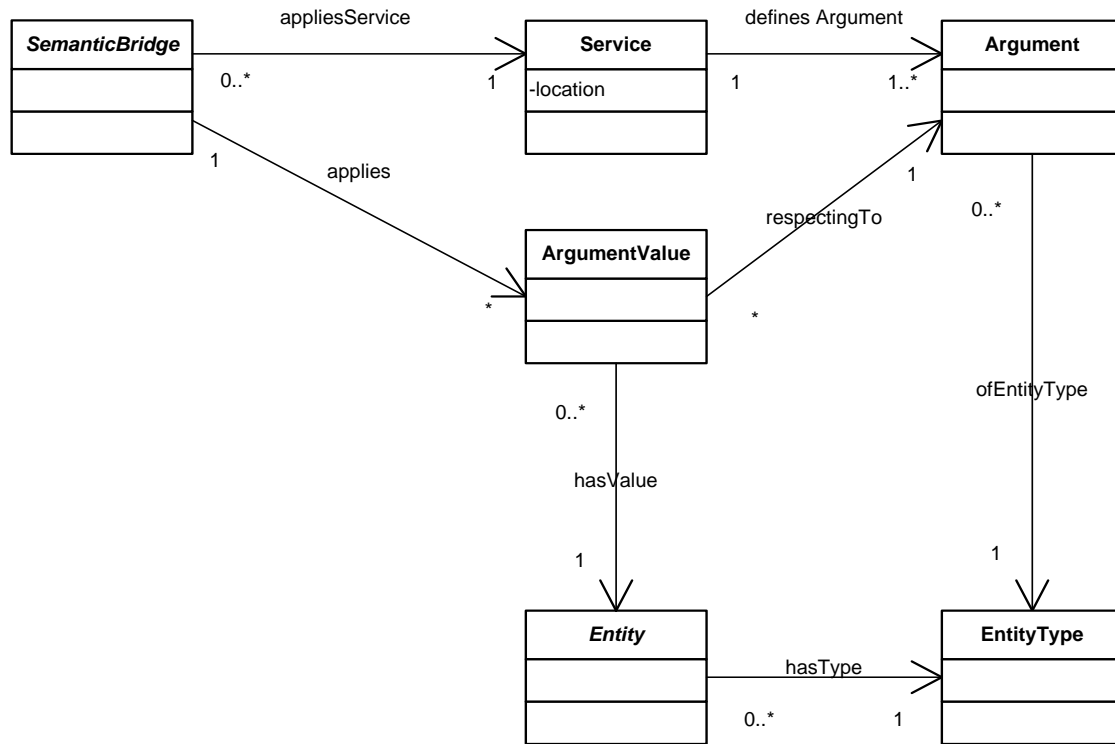


Figure 5.4 – UML representation of the SemanticBridge and Service conceptual relation

The clear separation of competences has three main advantages:

- It allows the separate evolution of semantic bridges upon the entities and transformation dimensions;
- It allows the enhancement of services according to operation and arguments, with minimal or no consequences to the semantic bridges specification;
- It allows the incorporation of new functionalities into Services with no consequences to the semantic bridges specifications. This possibility will be further exploited in other phases of the ontology mapping process, as described in Chapter 7.

The SBO component described so far is able to limitedly answer to the requirements raised by the Transformation and Cardinality dimensions of semantic relations. While insufficient to support all other requirements, it provides a supportive starting point to further and finer characterization of these and other SBO entities.

5.4.2 Service

The concept of Service is formally described in the scope of semantic bridging and execution phases as a tuple in the form of:

$$S := (Location, Args^s, Args^t, \alpha^s, \alpha^t) \in \mathcal{T}$$

where:

- \mathcal{T} is the set of all Services available in the ontology mapping system;
- *Location* corresponds to the necessary information to locate and access the transformation Service capabilities during the ontology mapping process;
- $Args^s$ and $Args^t$ are, respectively, the set of source and target Arguments composing the signature of the Service;
- α^s is the function that associates an *EntityType* with each and every source Argument:

$$\alpha^s : Args^s \rightarrow EntityType$$

- α^t is the function that associates an *EntityType* with each and every target Argument:

$$\alpha^t : Args^t \rightarrow EntityType$$

- *EntityType* is the set of entity types allowed in Services, corresponding to the union of:
 - Ontology concepts;
 - Paths, which are fully contextualized ontology properties (further described in 5.4.4);
 - Literals (constants);
 - Arrays of Paths and Arrays of Literals, which are collections of Paths and Literals (further described in 5.4.6)

Example 5.8 – Definition (instantiation) of a Service

Consider the CopyAttribute Service, capable to copy instances of a source ontology attribute into instances of a target ontology attribute.

$$S_{CopyAttribute} = \left(\text{"pt.ipp.isep.gecad.mafra.services.transformations.CopyAttribute"}, \right. \\ \left. Args_1^s, Args_1^t, \alpha_1^s, \alpha_1^t \right)$$

$$Args_1^s = \{sourceAttribute\}$$

$$Args_1^t = \{targetAttribute\}$$

$$\alpha_1^s = \{(sourceAttribute, attributePath)\}$$

$$\alpha_1^t = \{(targetAttribute, attributePath)\}$$

5.4.3 Semantic Bridge

The SemanticBridge concept conveys all necessary information from the source semantic bridging phase to the execution phase. According to the semantic relations characterization (5.2) and to previously sections, the SemanticBridge concept is formally defined as:

$$B := (\mathcal{E}^s, \mathcal{E}^t, \mathcal{Q}, \mathcal{S}, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{K})$$

where:

- \mathcal{E}^s and \mathcal{E}^t are respectively the source and target set of Entities, where Entities have an associated type (*EntityType*);
- \mathcal{Q} is a set of constants elements. Every constant element is either a single constant (Literal) or an Array of Literals (refer to 5.4.6);
- S represents the Service applied in the SemanticBridge;
- ϕ^s and ϕ^t are the functions that associate respectively the source and target entities to the parameters of the Service (each entity can be associated with more than one argument):

$$\phi^s : \mathcal{E}^s \rightarrow 2^{Args^s} \setminus \{\emptyset\}$$

$$\phi^t : \mathcal{E}^t \rightarrow 2^{Args^t} \setminus \{\emptyset\}$$

This function corresponds to the *ArgumentValue* class instances of Figure 5.4. Because Service is responsible for the definition and characterization of its own arguments, and because a *SemanticBridge* is related to only one Service, the cardinality of the *SemanticBridge* is in fact stated by the associated Service;

- δ^s and δ^t are the relations that associates constants with respectively source and target arguments of the transformation Service:

$$\delta^s : \mathcal{Q} \rightarrow 2^{Args^s} \setminus \{\emptyset\}$$

$$\delta^t : \mathcal{Q} \rightarrow 2^{Args^t} \setminus \{\emptyset\}$$

- \mathcal{K} is a set of *ConditionExpressions* that constrain the execution of the *SemanticBridge* according to the knowledge base instances. *ConditionExpressions* are further described in 5.4.5.

The formal definition of ontology (3.3) and the analysis of the entity type dimension of semantic relations (5.2.1) distinguish between concept and property entities. Following the same approach, *SemanticBridge* class is specialized into *ConceptBridge* and *PropertyBridge* classes (Figure 5.5).

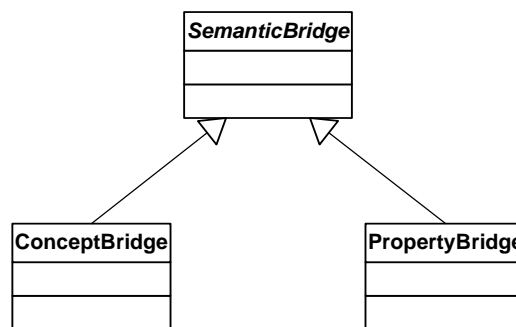


Figure 5.5 – UML representation of the SBO taxonomy of semantic bridges

5.4.3.1 Concept Bridge

ConceptBridge class represents the semantic relation between source and target ontology concepts. Semantically, this bridge means that each instance of the source concept gives rise to a new instance of the target concept. ConceptBridge specializes the SemanticBridge concept in two characteristics:

- The cardinality of a ConceptBridge is always 1:1, which means that exactly one source concept is semantically related to exactly one target concept. In order to semantically relate the same source concept with many different target concepts, multiple ConceptBridges should be defined (refer to Chapter 6);
- The transformation process executed in transforming concept instances into target concept instances is always the same, and corresponds to the CopyInstance Service. As consequence, the transformation Service specification in ConceptBridges can be made implicit.

Accordingly, a ConceptBridge is formally defined as:

$$B^C := (c^s, c^t, Q, \delta^s, \delta^t, \mathcal{K}) \in \mathcal{B}^C$$

where:

- \mathcal{B}^C is the set of all ConceptBridges;
- $c^s \in \mathcal{C}^s \subseteq \mathcal{E}^s$;
- $c^t \in \mathcal{C}^t \subseteq \mathcal{E}^t$;
- Q, δ^s, δ^t and \mathcal{K} are defined as for SemanticBridge.

The following functions are available:

- $sConcept : \mathcal{B}^C \rightarrow \mathcal{C}$, returns the source ontology concept of the ConceptBridge;
- $tConcept : \mathcal{B}^C \rightarrow \mathcal{C}$, returns the target ontology concept of the ConceptBridge.

5.4.3.2 Property Bridge

PropertyBridge class represents the semantic relation between sets of source and target ontologies properties. Semantically, this bridge means that the set of source properties instances are transformed into a set of target properties instances. Unlike ConceptBridge, the transformation occurring between source and target properties vary enormously. As consequence, the associated Service has to be explicitly stated in the PropertyBridge. PropertyBridge is formally defined as the following tuple:

$$B^P := (\mathcal{W}^s, \mathcal{W}^t, Q, S, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{K}) \in \mathcal{B}^P$$

where:

- \mathcal{B}^P is the set of all PropertyBridges;
- $\mathcal{W}^s \subseteq \mathcal{E}^s$ is the set of source ontology Paths (refer to 5.4.4);
- $\mathcal{W}^t \subseteq \mathcal{E}^t$ is the set of target ontology Paths;
- \mathcal{Q} , S , ϕ^s , ϕ^t , δ^s , δ^t and \mathcal{K} are defined as for SemanticBridge.

Notice that the associated Service is implicitly responsible for several characteristics of the bridge, in particular:

- The type of the Entities related through the bridge. A concatenation Service for example, requires string attributes from both source and target ontology, while a Service that would create relations between target instances requires relations as target ontology entities;
- The cardinality of the PropertyBridge. For example, the concatenation Service states an n:1 cardinality while the split Service states 1:n cardinality.

Unlike ConceptBridge that directly applies ontology concepts as their arguments, PropertyBridges applies ontologies properties in the context of certain ontology domain (a concept) and range (a concept or an attribute). This particularity requires other type of entity specification.

5.4.4 Path

According to the ontology model (3.3), properties define their own domain and range concepts. Additionally, the ontology model does not prevent the same property to have multiple domain and range concepts. This feature allows that certain instances use the same ontology property to relate two distinct types of property instances.

Example 5.9 – Multiple ranges of properties

The following KB, first presented in 3.3, depicts this feature. The `hasName` property in the domain of both `Researcher` and `Institution`:

$$\begin{aligned}
 \mathcal{KB}_1 &= (\mathcal{O}_1, \mathcal{I}_1, \text{inst}\mathcal{C}_1, \text{inst}\mathcal{P}_1) \\
 \mathcal{O}_1 &= (\mathcal{C}_1, \text{is_}a_1, \mathcal{P}_1, \sigma_1) \\
 \mathcal{C}_1 &= \{ \text{Researcher}, \text{Institution} \} \\
 \text{is_}a &= \{ \mathcal{C}_1, \mathcal{C}_1, \{ \} \} \\
 \mathcal{P}_1 &= \{ \text{hasName}, \text{researchesIn}, \text{inCity} \} \\
 \sigma_1 &= \left\{ \begin{array}{l} \text{hasName}(\text{Researcher}, \text{Literal}), \text{hasName}(\text{Institution}, \text{Literal}), \\ \text{researchesIn}(\text{Researcher}, \text{Institution}), \text{inCity}(\text{Institution}, \text{Literal}) \end{array} \right\}
 \end{aligned}$$

Due to this central role properties play in the modeling process and data model, and besides the object-oriented modeling capabilities, ontologies of this kind are (also) categorized as property-centric ontologies.

This ontology feature must be included in the Semantic Bridging Ontology such the domain and range of the property, are specified as intended in each specific context.

5.4.4.1 Step

This feature is achieved through the Step concept, also referred as statement, which is a tuple in the form of:

$$s := (\textit{subject}, \textit{predicate}, \textit{object}, \textit{direction}) \in \mathcal{S}$$

where:

- \mathcal{S} is the set of all Steps;
- $\textit{subject} \in \mathcal{C}$ is the domain of $\textit{predicate}$;
- $\textit{predicate} \in \mathcal{P}$ is the ontology property;
- $\textit{object} \in \mathcal{C} \cup \{\textit{Literal}\}$ is the range of the ontology property, which can be either an ontology concept or Literal;
- $\textit{direction}$ is the forward or backward value defining the direction the predicate is read, i.e. “from subject” or “to subject” (refer to 5.4.4.3).

Every Step must represent a valid relation in the ontology, such that:

$$\textit{subject} \in \text{domain}(\textit{predicate}) \wedge \textit{object} \in \text{range}(\textit{predicate})$$

Complementarily, three functions are available:

- $\textit{subject} : \mathcal{S} \rightarrow \mathcal{C}$ gives the ontology concept playing the role of subject in the Step;
- $\textit{predicate} : \mathcal{S} \rightarrow \mathcal{P}$ gives the ontology property playing the role of predicate in the Step;
- $\textit{object} : \mathcal{S} \rightarrow \mathcal{C} \cup \{\textit{Literal}\}$ returns the concept or Literal playing the role of object in the Step.
- However, Steps are not directly applied in SemanticBridges but grouped together in Paths.

5.4.4.2 Path

In fact, due to distinct ontological decisions made in modeling ontologies, semantically equivalent entities from two ontologies are often located in different levels of the ontologies structure.

Example 5.10 – Path is required between two structurally different ontologies

In the ontology mapping scenario of Figure 5.6, O1:Researcher.researchesIn.Institution.inCity.Literal is semantically related to O2:Person.address.Literal. This relation semantically means that both attributes (inCity and address) are semantically related, but only when inCity is accessed through the fully qualify Path (O1:Researcher.researchesIn.Institution.inCity.Literal). In fact, O1:Institution.inCity.Literal is not directly semantically related to O2:Person.address.Literal because no semantic relation exists between O1:Institution and O2:Person without the researchesIn relation.

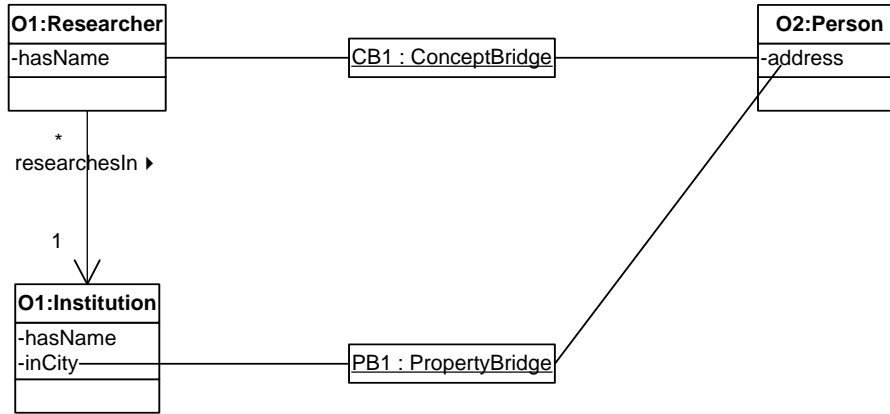


Figure 5.6 – Path between two structurally different ontologies

Addressing properties in distinct levels of the ontology in respect to a specific entry class may be necessary to overcome semantic heterogeneity.

To fulfill this requirement the Path concept is proposed. The Path concept corresponds to a non-empty list³⁵ of Steps:

$$W := [s_1, s_2, \dots, s_n] \in \mathcal{W}$$

such that:

- \mathcal{W} is the set of all Paths;
- $s_i \in \mathcal{S}$;
- $\text{lenght} : \mathcal{W} \rightarrow \mathbb{N}^+$ is the function that gives the (positive integer) number of Steps of the Path;
- $\text{entry} : \mathcal{W} \times \mathbb{N}^+ \rightarrow \mathcal{S}$ is the function that returns the n th entry in the Path.

Because Path represents a set of valid relations between multiple concepts, the subject of certain Step in the Path should be the same object of the previous Step of the Path. This constraint is formally defined as:

$$\forall s_i, s_{i+1} \in \mathcal{S}, W \in \mathcal{W} \left((\text{entry}(W, i) = s_i \wedge \text{entry}(W, i+1) = s_{i+1}) \Rightarrow \text{object}(s_i) = \text{subject}(s_{i+1}) \right)$$

Example 5.11 – Path representation

The path mentioned in previous example (O1:Researcher.researchesIn.Institution.inCity.Literal), is therefore represented as:

$$W_1 = \left[\begin{array}{l} (O1: Researcher, O1: researchesIn, O1: Institution, forward), \\ (O1: Institution, O1: inCity, Literal, forward) \end{array} \right].$$

³⁵ A list is a (possibly empty) ordered collection (sequence) of elements, referred as entries, in which repetition is allowed. Lists are seen as sets in which duplicated entries are allowed and order of entries do matters.

Furthermore, Paths are differentiated according to the range of their last Step:

- Attribute Paths, if $\text{object}(s_{\text{length}(W)}) \in \{Literal\}$
- Relation Paths, if $\text{object}(s_{\text{length}(W)}) \in \mathcal{C}$

No cyclic Paths are allowed, but the same Step can be present more than once in the Path.

5.4.4.3 Directionality

Properties are directional predicates between the subject and the object: subject “relates to” object. However, due to distinct semantic decisions, it often occurs in ontology mapping scenarios that certain source ontology property semantically relates to a target ontology property that is exactly its inverse property.

Example 5.12 – Ontology mapping scenario requiring inverse Path

Consider the ontology mapping scenario of Figure 5.7 in which ontology O1 and ontology O2 are being semantically bridged. O1 corresponds to the previous presented ontology (3.3) and O2 is a similar ontology concerning the same real-world domain.

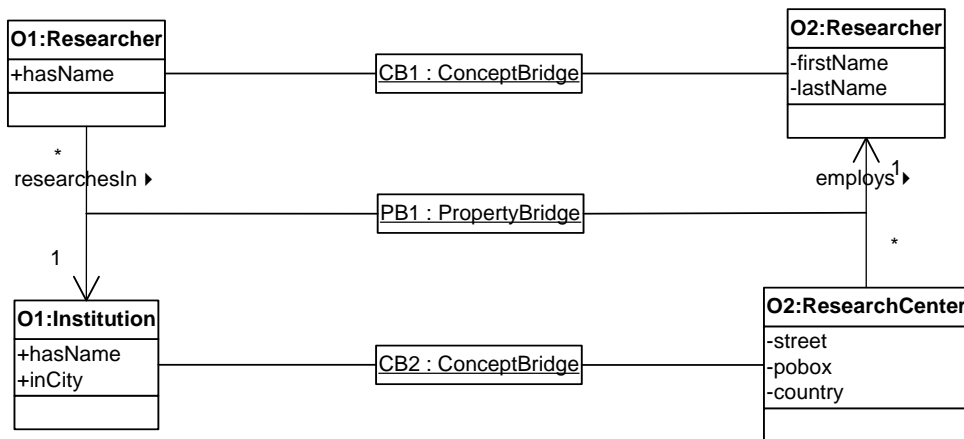


Figure 5.7 – Ontology mapping scenario dealing with inverse properties

While O1:Researcher bridges to O2:Researcher and O1:Institution bridges to O2:ResearchCenter, the O1:Researcher.researchesInInstitution property corresponds to the inverse property of O2:ResearchCenter.employs.Researcher.

In order to overcome these semantic differences, it is necessary to (virtually) address properties in inverse direction, corresponding during execution phase, to inversely query the knowledge base: which are the subjects that “relate to” the object. In the case of the previous example, it would correspond to: “which are the O2:ResearchCenters in which O2:Researcher is employee”.

Inverse properties are defined through Steps (inverse Steps or backward Steps) in which the *direction* parameter is stated to backward.

Example 5.13 – Definition of a backward Step

From the previous example, the inverse Step would be represented as the Step $s_1 = (O2: ResearchCenter, O2: employs, O2: Researcher, backward)$.

Notice that the direction is asserted to the property (Step) and not to the Path, allowing that the same Path combines simultaneously properties read in forward and backward direction. However, in case Paths include an inverse Step, they are also referred as inverse Paths (or backward Paths).

Example 5.14 – Definition of a backward Path

The Path corresponding to Example 5.12 would be defined as:

$$W_2 = [(O2: ResearchCenter, O2: employs, O2: Researcher, backward)]$$

5.4.4.4 Alternative notation

In order to simplify previous notation, from now on, Path is specified through an easier notation that assumes two distinct forms:

- *Subject / Predicate / Object*, if the direction value of the Path is forward;
- *Subject \ Predicate \ Object*, if the direction value of the Path is backward.

Example 5.15 – Definition of Paths using the simplified notation

Accordingly, Path from Example 5.11 and Example 5.14 would be defined as:

$$W_1 = O1: Researcher / O1: researchesIn / O1: Institution / O1: inCity / Literal$$

$$W_2 = O2: ResearchCenter \ O2: employs \ O2: Researcher$$

Furthermore, the reference to the ontology can be omitted in cases it is irrelevant or no ambiguities arises.

5.4.4.5 Outlook of Path

Paths are a very powerful mechanism to specify ontology entities into SemanticBridges and query the knowledge base during the execution phase. While the Path mechanism under the point-of-view of Semantic Bridging has been analyzed in this section, the Path operation in querying and iterating the knowledge base is addressed in Chapter 6.

5.4.5 Condition Expression

It often occurs that the SemanticBridge depends not only on the set of source and target ontology entities, but also on the instances of such ontology entities. ConditionExpressions provide the mechanism to specify semantic constraints upon SemanticBridges. ConditionExpressions are then instantiated with ontology entities instances in execution phase and evaluated accordingly.

ConditionExpression is a Boolean expression of comparison expressions that compares instances with instances or constants (Literals), through a variety of comparison operators. Using the BNF notation, ConditionExpression corresponds to:

```

<condition_expression> ::= <and> | <or> | <xor> | <not> | <comparison>
<and> ::= and ( <condition_expression> { "," <condition_expression> } )
<or> ::= or( <condition_expression> { "," <condition_expression> } )
<xor> ::= xor( <condition_expression> { "," <condition_expression> } )
<not> ::= not( <condition_expression> )
<comparison> ::= <operand_1> <OPERATOR> <operand_2>
<operand_1> ::= <PATH>
<operand_2> ::= <PATH> | <LITERAL>
    
```

- The <PATH> token corresponds to the Path entity described above (5.4.4), and its application in ConditionExpressions follows the same constraints defined then;
- The <OPERATOR> token corresponds to the comparison operator. Some of the envisaged comparison operators are represented in Table 5.5:

Table 5.5 – SBO comparison operators

Operator	Meaning
==	Op1 “equal to” Op2
<	Op1 “less than” Op2
=<	Op1 “equal or less than” Op2
>=	Op1 “equal or greater than” Op2
>	Op1 “greater than” Op2
Match	Op1 “matches regular expression” Op2
Like	Op1 “string contains string” Op2 (<i>a la SQL</i>)
Cardinality	Op1 “cardinality is” Op2
MaxCardinality	Op1 “cardinality is less than” Op2
MinCardinality	Op1 “cardinality is greater than” Op2

- The <LITERAL> token corresponds to the Literal ontology entity.

Example 5.16 – Definition of a ConditionExpression

Considering the knowledge base presented in section 3.3, a valid condition expression would be:

$$K_1 = \text{and} \left(\begin{array}{l} \text{Researcher} / \text{researchesIn} / \text{Institution} / \text{hasName} == \text{"GECAD - ISEP"}, \\ \text{Researcher} / \text{researchesIn} / \text{Institution} \text{ Cardinality } 1 \end{array} \right)$$

This condition requires that the instance of Researcher researches in the Institution with name “GECAD-ISEP” and only there.

The following ConditionExpression is always true:

$$K_2 = \text{Researcher} / \text{researchesIn} / \text{Institution} == \text{Researcher} / \text{researchesIn} / \text{Institution}$$

5.4.6 Array

Every Service defines a specific number of arguments and respective types, to which the SemanticBridge applying the Service must conform to. In case an extra argument is required in the SemanticBridge, a new Service should be developed and applied.

While the transformation requirements are unpredictable and therefore the development of new Services will be always necessary, the problem can be minimized by the generalization of transformation Services. Generalization is drastically restricted by the function dimension (5.2.2.1), which among other affects the type and number of arguments. Nevertheless, certain functions can be generalized, especially respecting the number of arguments.

Example 5.17 – Generalization of Services parameters

Consider the Concatenation Service in which two source attribute instances are concatenated into one target attribute instance (e.g. `concatenation("Ontology", "Mapping")`).

While typical concatenation process is specified for two attribute instances, the process is extensible for a variable number of attribute instances (e.g. `concatenation("Ontology", " ", "Mapping") == "Ontology Mapping"`).

Instead of developing a new Service capable to concatenate a specific number of attributes, the Concatenation Service can be generalized such that a variable number of attribute instances is concatenated.

The Array construct allows the generalization of Services according to the cardinality dimension. Array is an ordered set of mapping entities of the same type, but unlike sets in which order does not matter, or in a list in which elements are typically accessed successively, in array the order matters and elements are accessed in an arbitrary order, as might be required by specific Services. Concepts, Paths, ConditionExpressions and Literals can be grouped into Array and applied in the SemanticBridge as a single argument to the Service. The following types of Array are valid in SBO:

- Array of Paths;
- Array of ConditionExpressions;
- Array of Literals.

Because Services are more generic, specification of SemanticBridges is easier and more durable.

Example 5.18 – Definition of Service using Arrays

Instead of choosing between Services such as Concatenation/3³⁶ (concatenation of two source ontology attribute into one target ontology attribute), Concatenation/4 (concatenation of three source ontology attributes into one target ontology attribute), one might use Concatenation/N capable to concatenate N-1 source ontology attributes into one target ontology attribute. The Concatenation/N Service would be described as:

³⁶ Notation used in many programming languages (Prolog, Lisp, Erlang) to characterize the arity (number of arguments) of the function or procedure.

$$S_{Concatenation} = \left(\begin{array}{l} \text{"pt.ipp.isep.gecad.mafra.services.transformations.Concatenation",} \\ \text{Args}_2^s, \text{Args}_2^t, \alpha_2^s, \alpha_2^t \end{array} \right)$$

$$\text{Args}_2^s = \{\text{sourceAttributes}, \text{separators}\}$$

$$\text{Args}_2^t = \{\text{targetAttribute}\}$$

$$\alpha_2^s = \{(\text{sourceAttributes}, \text{arrayOfAttributePaths}), (\text{separators}, \text{arrayOfLiterals})\}$$

$$\alpha_2^t = \{(\text{targetAttribute}, \text{attributePath})\}$$

The separators source argument allows the definition of a set of Literals to concatenate between every pair of source ontology attribute instance.

5.4.7 Ontology Mapping Document

The core concepts of SBO have been addressed in previous sections. However, the ontology mapping domain of knowledge is insufficiently described, and thus, even the described concepts lack semantics. In particular, it is necessary to contextualize and inter-relate SBO concepts and provide mechanisms to finer constrain and explicitly describe semantic relations in order to reach the intended ontology mapping execution.

The Ontology Mapping Document provides these necessary elements to inter-relate SBO concepts, constraining their meaning and interpretation possibilities. Ontology Mapping Document is formally defined as the following 11-ary tuple:

$$\mathcal{M} := \left(\mathcal{O}^s, \mathcal{O}^t, \mathcal{T}, \mathcal{B}^C, \mathcal{B}^P, \mathcal{A}^{\mathcal{B}^C}, \mathcal{A}^{\mathcal{B}^P}, \perp^{\mathcal{B}^C}, \perp^{\mathcal{B}^P}, \diamond, \prec \right)$$

where:

- \mathcal{O}^s is the source ontology;
- \mathcal{O}^t is the target ontology;
- \mathcal{T} is the set of available transformation Services;
- \mathcal{B}^C is the set of ConceptBridges holding between \mathcal{O}^s and \mathcal{O}^t concepts;
- \mathcal{B}^P is the set of PropertyBridges holding between \mathcal{O}^s and \mathcal{O}^t defined Paths;
- $\mathcal{A}^{\mathcal{B}^C}$ is the set of containers, named AlternativeBridges-of-ConceptBridges that group together mutually disjoint ConceptBridges;
- $\mathcal{A}^{\mathcal{B}^P}$ is the set of containers, named AlternativeBridges-of-PropertyBridges, that group together mutually disjoint PropertyBridges;
- $\perp^{\mathcal{B}^C} : \mathcal{A}^{\mathcal{B}^C} \rightarrow 2^{\mathcal{B}^C}$ is the function that associates ConceptBridges with AlternativeBridges-of-ConceptBridges;
- $\perp^{\mathcal{B}^P} : \mathcal{A}^{\mathcal{B}^P} \rightarrow 2^{\mathcal{B}^P}$ is the function that associates PropertyBridges with AlternativeBridges-of-PropertyBridges;

- $\diamond: \mathcal{B}^C \rightarrow 2^{\mathcal{B}^P \cup \mathcal{A}^{\mathcal{B}^P}}$ is the function that relates PropertyBridges with ConceptBridges.
- $\prec \subseteq \mathcal{B}^C \times \mathcal{B}^C$ is a reflexive, acyclic, anti-symmetric and transitive relation between ConceptBridges, which corresponds to the hierarchical relation between ConceptBridges, referred as “sub bridge of”.

Figure 5.8 depicts these relations using the UML notation.

Next sections describe $\mathcal{A}^{\mathcal{B}^C}$, $\mathcal{A}^{\mathcal{B}^P}$ concept, together with the $\perp^{\mathcal{B}^C}$, $\perp^{\mathcal{B}^P}$ and \diamond functions and the \prec relation.

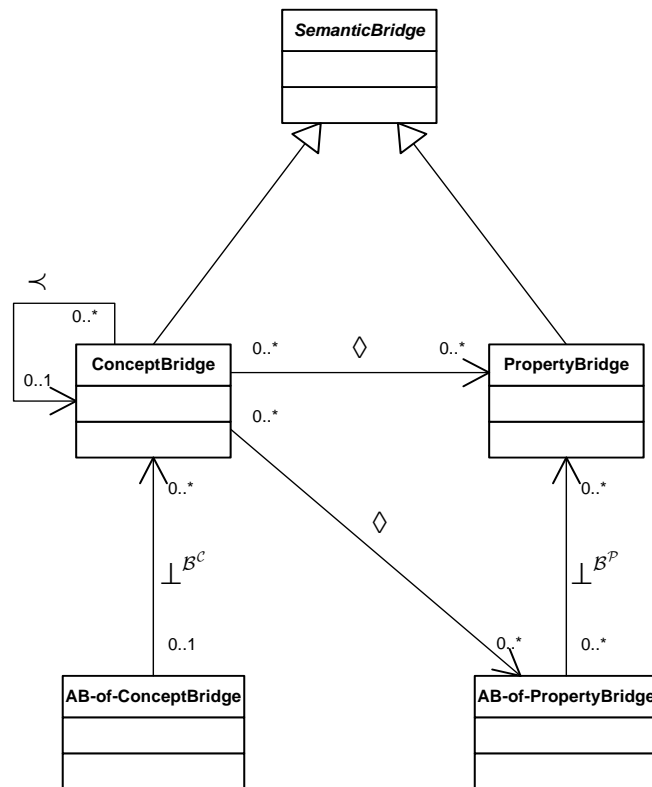


Figure 5.8 – UML representation of the SBO relations between SemanticBridges

5.4.7.1 Alternative Bridges of ConceptBridges

The $\perp^{\mathcal{B}^C}: \mathcal{A}^{\mathcal{B}^C} \rightarrow 2^{\mathcal{B}^C}$ function is responsible for the specification of disjoint relations between PropertyBridges into AlternativeBridge-of-ConceptBridges ($\mathcal{A}^{\mathcal{B}^C}$). This relation between ConceptBridges arises from the need to conditionally transform the instances of one source concept into instances of a set of target concept. In such cases, one ConceptBridge is defined for each pair of the source ontology concept and target ontology concept. These ConceptBridges are then grouped together into an AlternativeBridge-of-ConceptBridges. Every ConceptBridge is free to define ConditionExpressions that determine its execution conditions. AlternativeBridges

prevents the execution of more than one of disjoint ConceptBridges, even if multiple ConceptBridges are conditionally allowed.

Each ConceptBridge can be $\perp^{\mathcal{B}^c}$ -related once:

$$\forall cb \in \mathcal{B}^c, \forall a_1, a_2 \in \mathcal{A}^{\mathcal{B}^c} \quad \perp^{\mathcal{B}^c}(cb, a_1) \wedge \perp^{\mathcal{B}^c}(cb, a_2) \Rightarrow a_1 = a_2$$

Besides its execution features, by preventing the execution of more than one of the ConceptBridges, AlternativeBridge-of-ConceptBridge becomes a modeling primitive that enhances the semantics and readability of the ontology mapping document, by explicitly stating the disjoint relation between ConceptBridges.

Example 5.19 – AlternativeBridges of ConceptBridges

Consider the ontology mapping scenario of Figure 5.2 (page 84). Two mutually exclusive ConceptBridges should be defined from O1:Person to O2:Man and O2:Woman. Explicitly stating that these two ConceptBridges are mutually exclusive states that an instance of O1:Person is either O2:Man or O2:Woman. This example is illustrated in 5.5.

5.4.7.2 Alternative Bridges of PropertyBridges

The $\perp^{\mathcal{B}^p} : \mathcal{A}^{\mathcal{B}^p} \rightarrow 2^{\mathcal{B}^p}$ function is responsible for the specification of disjoint relation between PropertyBridges, into AlternativeBridge-of-PropertyBridges ($\mathcal{A}^{\mathcal{B}^p}$). Similarly to the disjoint relation between ConceptBridges, a set of PropertyBridges is often mutually disjoint. While ConditionExpressions are specified for every SemanticBridge, ConditionExpression might not be sufficient to guaranty the disjointedness. Besides guaranty disjointedness, AlternativeBridge-of-PropertyBridges is able to explicitly state constraints, promoting readability and explicit semantics.

Unlike ConceptBridges whose context is the Ontology Mapping Document (\mathcal{M}), specification and execution context of PropertyBridges are ConceptBridges. Also, because every PropertyBridge can be associated (\diamond -related) with multiple ConceptBridges (see 5.4.7.3), PropertyBridge is ultimately contextualized by the ConceptBridge in which it is being-related or executed. As consequence, it is possible that a PropertyBridge is disjoint in a certain context (ConceptBridge) but not in another. Accordingly, every PropertyBridge can be $\perp^{\mathcal{B}^p}$ -related more than once.

5.4.7.3 Relation between ConceptBridges and PropertyBridges

A SemanticBridge is specialized into ConceptBridge and PropertyBridge. The most important difference between these two classes is the type of Service each one applies. While the applied Service in ConceptBridge is always responsible for the creation of a target concept instance, the Service in PropertyBridge varies according to the semantic relations between ontologies properties.

This clear distinction is very important for the characterization of their inter-relations. In fact, notice that ConceptBridges are responsible for the creation of instances of target ontology concepts, which in turn will serve as containers or placeholders for instances of target ontology properties resulting from the execution of PropertyBridges. This means that concept instances are composed by properties instances. As consequence, ConceptBridges and PropertyBridges are closely inter-related and interdependent. The \diamond function represents this inter-relation.

Through the \diamond function, PropertyBridges are contextualized into ConceptBridges. During semantic bridging phase, this relation means that the properties of the source concept correspond to the target concept properties according to the PropertyBridge. At execution time, the \diamond -related PropertyBridge will be executed for all properties instances defined in the source concept instances, giving rise to the target concept properties instances.

Because AlternativeBridges-of-PropertyBridges ($\mathcal{A}^{\mathcal{B}^P}$) are composed of PropertyBridges only (which semantically corresponds to the execution of one of a set of PropertyBridges), AlternativeBridges-of-PropertyBridges are considered as PropertyBridge. As defined, the \diamond -relation relates ConceptBridges to both PropertyBridges and AlternativeBridges-of-PropertyBridges.

Moreover, the PropertyBridges defined disjoint in the scope of an AlternativeBridge-of-PropertyBridge are not directly \diamond -related with ConceptBridge, but only through the AlternativeBridge-of-PropertyBridge it is \diamond -related.

The \diamond function has consequence on the properties allowed in PropertyBridges. In fact, properties that are not accessible from the concepts semantically related in the ConceptBridge can not be semantically related in such context. If PropertyBridges are \diamond -related with certain ConceptBridge, it is mandatory that the Paths defined in PropertyBridges³⁷ all have the ConceptBridge concepts as root concept³⁸. This constraint is formally represented as the following conjunction:

$$\begin{aligned}
 cb &= (c^s, c^t, \mathcal{Q}_{cb}, \delta_{cb}, \mathcal{K}_{cb}) \in \mathcal{B}^C \\
 pb &= (\mathcal{W}^s, \mathcal{W}^t, \mathcal{Q}, S, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{A}_{pb}, \mathcal{K}_{pb}) \in \mathcal{B}^P \\
 \forall W^s \in \mathcal{W}^s, s^s = \text{entry}(W^s, 1) \quad \diamond(cb, pb) &\Rightarrow \text{subject}(s^s, c^s) \wedge \\
 \forall W^t \in \mathcal{W}^t, s^t = \text{entry}(W^t, 1) \quad \diamond(cb, pb) &\Rightarrow \text{subject}(s^t, c^t)
 \end{aligned}$$

Additionally, the \diamond function further constrains the length of target Paths to 1:

$$\forall W \in \mathcal{W}^t \quad \text{lenght}(W) == 1$$

³⁷ Properties are specified in PropertyBridges through Paths.

³⁸ Root concept is the ontology concept playing the role of subject in the first Step of the Path.

Besides not representing a conceptual requirement but instead an implementation decision, four important advantages have been identified, contributing to maintain and promote this constraint:

1. Permits to ignore the execution order of SemanticBridges. If a multi-step target Path is specified, it would be necessary to previously execute the PropertyBridges that create the relation between instances;
2. As consequence, it prevents recursive dependencies between SemanticBridges;
3. Promotes modularization, systematization and error detection. If the application of a multi-step target Path is believed necessary, then it is necessary to \diamond -relate the PropertyBridge with a ConceptBridge, such:

$$\forall W \in \mathcal{W}^t \quad \text{tConcept}(cb) == \text{subject}(W, \text{lenght}(W))$$

and change Paths, such:

$$\forall W \in \mathcal{W}^t \quad W = [s \mid s = \text{entry}(W, \text{lenght}(W))]$$

4. Evolution procedures are simplified. Since fewer dependencies exist between SemanticBridges, fewer number of propagation changes arise between SemanticBridges.

5.4.7.4 Hierarchy of ConceptBridges

The \prec -relation (subBridgeOf) is the SBO correspondence to the ontology construct *is_a* (subclass of) provided in object-oriented ontology models (such as the one suggested in 3.3). Supporting the subBridgeOf modeling construct, SBO provides:

- Capabilities to better model the semantic relations, including the generalization/specialization of semantic relations, which has been proven to be a very powerful modeling construct in knowledge representation languages;
- Capabilities to model SemanticBridges according to the hierarchy of concepts.

Any ConceptBridge may be defined subBridgeOf another ConceptBridge if its source and target concepts are sub concepts (*is_a*) of respectively the source and target concept of the envisaged super-bridge. Additionally, the relation is valid in case either the source concepts or the target concepts are the same. This constraint is formally defined by:

$$\begin{aligned} & \forall cb_1, cb_2 \in \mathcal{B}^c \\ & \quad \prec(cb_2, cb_1) \\ \Rightarrow & \quad \text{sConcept}(cb_1, c_1^s) \wedge \text{tConcept}(cb_1, c_1^t) \wedge \text{sConcept}(cb_2, c_2^s) \wedge \text{tConcept}(cb_2, c_2^t) \wedge \\ & \quad \left((is_a(c_2^s, c_1^s) \wedge is_a(c_2^t, c_1^t)) \vee (is_a(c_2^s, c_1^s) \wedge c_1^t == c_2^t) \vee (c_1^s == c_2^s) \wedge is_a(c_2^t, c_1^t) \right) \end{aligned}$$

Example 5.20 – Hierarchy of ConceptBridges

Figure 5.9 presents an ontology mapping scenario in which the hierarchical relation between ConceptBridges (\prec) is adopted according to the hierarchy of the ontology concepts. The represented \prec -relations are defined as:

$$\prec(CB2a, CB1), \prec(CB2b, CB1), \prec(CB3, CB2a) \text{ and } \prec(CB3, CB2b), \text{ or as the set } \prec := \{(CB2a, CB1), (CB2b, CB1), (CB3, CB2a), (CB3, CB2b)\}$$

Notice that $\prec(CB2a, CB1)$ and $\prec(CB2b, CB1)$ relationships are valid due to the $(c_1^s = c_2^s \wedge is_a(c_2^t, c_1^t))$ component of the constraint.

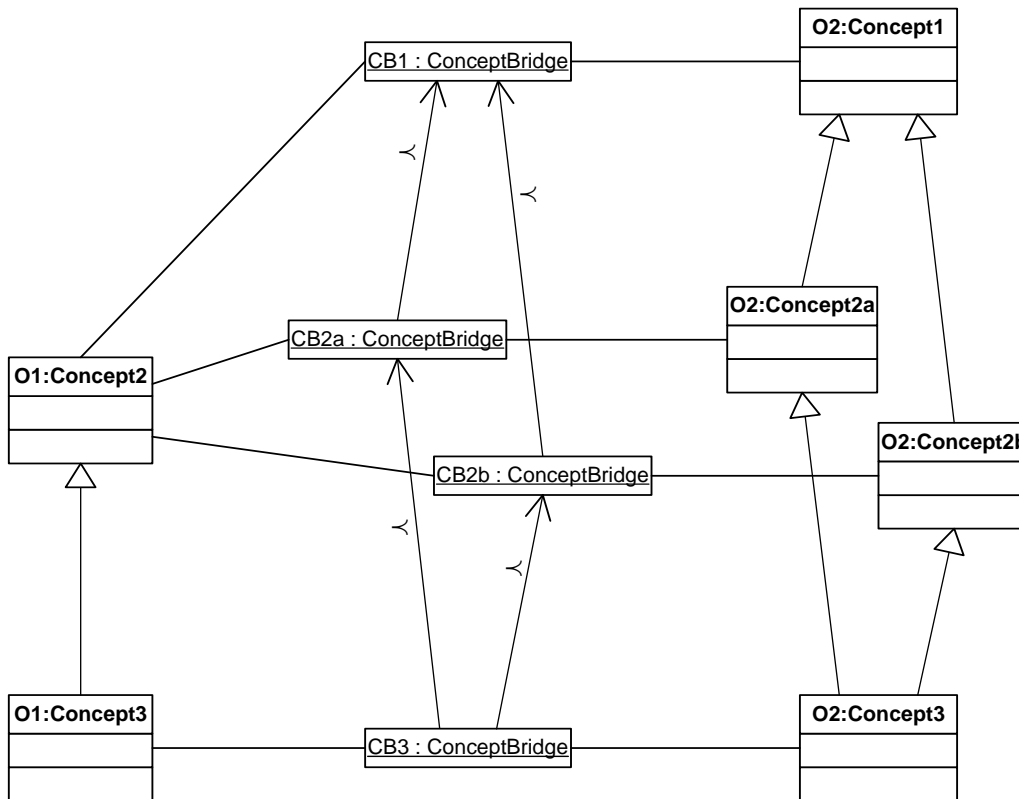


Figure 5.9 – Hierarchical relations between ConceptBridges

The fundamental consequence of this relation is the inheritance by sub-bridges of the PropertyBridges \diamond -related with super-bridges. As so, such PropertyBridges are automatically and implicitly \diamond -related to its sub-bridges. During execution phase, the PropertyBridges of the super-bridge with are executed in the scope of sub-bridges as any explicitly \diamond -related PropertyBridge.

Complementarily to the hierarchy of concepts, one very popular and useful constructs in OO representation languages is that of the “abstract concept”. Abstract concepts are characterized by the fact that no instances of such concepts are allowed. In some representation languages (e.g. RDFS, OIL, DAML, OWL) such primitive is not provided, but the modeling decision is yet adopted by the ontology engineer.

Such modeling decision, either implicit or explicit, assumes special relevance in the ontology mapping specification, since no instances should be created for an abstract target ontology concept. The ontology mapping representation language (SBO) must provide support for this ontology feature.

ConceptBridge and the CopyInstance Service are the candidates to accommodate this feature/parameter. Notice however that unlike ConceptBridges in which the Service is always the same, in PropertyBridges Services vary from case to case, as their specific parameters. Therefore, in order to follow and maintain a coherent specification of SBO, this very specific parameter is associated with the CopyInstance Service, and not with the ConceptBridge, which would require a new kind of element in the ConceptBridge definition. An example can be found on 5.5.6 and details about the CopyInstance Service can be found in Chapter 6.

This hierarchical inheritance mechanism, similar to object-oriented modeling, profits and provides the benefices recognized to the object-oriented modeling. In special, when applied to distributed, well-structured, inter-dependent ontologies, which are progressively adopted and supported in Semantic Web [Maedche *et al.*, 2003], it promotes modularity, reusability and readability of the ontology mapping document.

5.5 Example 5.21 – Semantic bridging annotated example

This section presents semantic bridging examples through the instantiation of SBO. While typical semantic relations tend to be subjective and ambiguous, the following examples use common sense ontologies and semantic relations between them.

Consider Figure 5.10 where excerpts³⁹ of two ontologies are represented in UML notation. In the left side, playing the role of source ontology is the Gedcom ontology (O1) [Gedcom] and in the right, playing the role of target ontology is the Gentology ontology (O2) [Gentology].

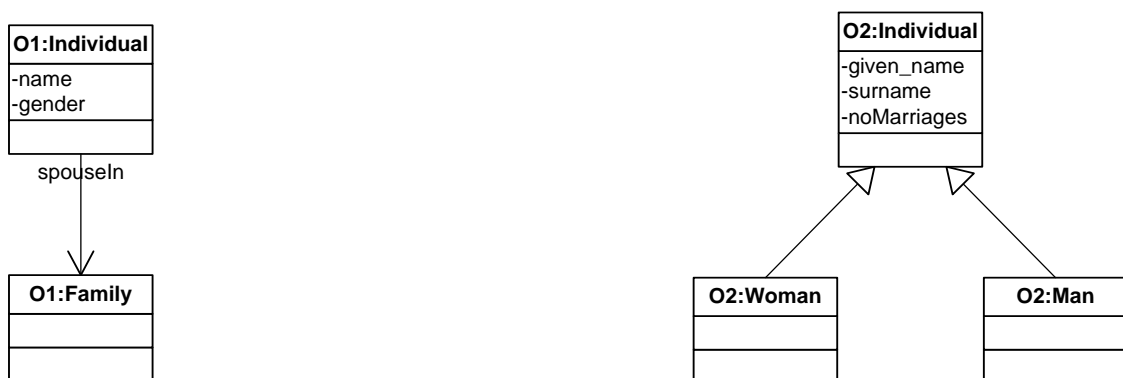


Figure 5.10 - Excerpt of Gedcom and Gentology ontologies, represented in UML notation

³⁹ For readability reasons, some concepts and properties of both ontologies are omitted.

5.5.1 Ontology Mapping Document

The primary element to define is the Ontology Mapping Document (in which both ontologies are explicitly related). Also other ontology mapping components are initially introduced:

$$\mathcal{M}_1 = \left(O1, O2, \mathcal{T}_1, \mathcal{B}_1^C, \mathcal{B}_1^P, \mathcal{A}_1^{\mathcal{B}^C}, \mathcal{A}_1^{\mathcal{B}^P}, \perp_1^{\mathcal{B}^C}, \perp_1^{\mathcal{B}^P}, \diamond_1, \prec_1 \right)$$

$$\mathcal{T}_1 = \{ CopyAttribute, CountProperties, Concatenation, Split \}$$

5.5.2 ConceptBridges

Both source and target ontologies define the concept Individual, which must be bridged. Because the target ontology entity in the semantic relation is a concept, the expected outcome of the semantic bridge are target ontology concept instances. The transformation/creation of target ontology concept instances is a competence of ConceptBridges. Accordingly, a ConceptBridge should be used between O1:Individual and O2:Individual:

$$Individual2Individual = \left(O1: Individual, O2: Individual, Q_{I2I}, \delta_{I2I}^s, \delta_{I2I}^t, K_{I2I} \right)$$

$$Q_{I2I} = \{ \}$$

$$\delta_{I2I}^s = \{ \}$$

$$\delta_{I2I}^t = \{ \}$$

$$K_{I2I} = \{ \}$$

The previous line defines a ConceptBridge, whose source ontology concept is O1:Individual and the target ontology concept is O2:Individual. Actually, O1:Individual instances will give rise to either O2:Woman or O2:Man instances. For that reason, two more ConceptBridges are necessary:

$$Indiv2Woman = \left(O1: Individual, O2: Woman, \{ \}, \{ \}, \{ \}, K_{I2W} \right)$$

$$Indiv2Man = \left(O1: Individual, O2: Man, \{ \}, \{ \}, \{ \}, K_{I2M} \right)$$

Therefore, \mathcal{B}_1^C can now be defined as:

$$\mathcal{B}_1^C = \{ Individual2Individual, Indiv2Man, Indiv2Woman \}$$

5.5.3 ConditionExpressions

Each of previous ConceptBridges is conditionally executed according to the value of O1:Individual/genre/Literal attribute of every O1:Individual being transformed: if the O1:Individual instance is masculine it semantically relates to O2:Man, otherwise it semantically relates to O2:Woman. Accordingly, both K_{I2M} and K_{I2W} are necessarily set to:

$$K_{I2W} = \{ O1: Individual / genre / Literal == "W" \}$$

$$K_{I2M} = \{ O1: Individual / genre / Literal == "M" \}$$

As previously described, ConditionExpression is a Boolean expression of comparison expressions, providing the means to specify very complex conditions. Even if this example does not require complex ConditionExpressions, it is strongly suggested to improve both K_{I2M} and K_{I2W} , in order to cover wider coding possibilities (e.g. “Male”, “Masculine”, “Female”, and “Feminine”). For example, one could further specify K_{I2M} to:

$$K_{I2M} = \left\{ \text{or} \left(\begin{array}{l} O1: \text{Individual} / \text{genre} / \text{Literal} == "M", \\ O1: \text{Individual} / \text{genre} / \text{Literal Match } "\wedge M | M *" \end{array} \right) \right\}$$

5.5.4 Disjoint bridges

The specification of ConditionExpressions in independent SemanticBridges might not be sufficient for controlling the execution of the ontology mapping. In fact, definition of ConditionExpressions ensures that the SemanticBridge is executed only if a condition holds, but it does not ensure that other bridges are not executed. This would not be a problem in the running example if the O1:Individual/genre/Literal for each instance of O1:Individual is either “F” or “M”. But, if by some reason, both values are specified, two O2:Individual instances will be created. This is considered a semantic error. Similar situation can additionally occur in case no sufficient conditions or ill-specified conditions are associated with one of the disjoint bridges.

In this particular case, because Indiv2Man and Indiv2Woman are ConceptBridges, the AlternativeBridge-of-ConceptBridges is applied:

$$\begin{aligned} \mathcal{A}_1^{B^c} &= \{ \text{ManOrWoman} \} \\ \perp_1^{B^c} &= \left\{ \left(\text{ManOrWoman}, \{ \text{Individual2Woman}, \text{Individual2Man} \} \right) \right\} \end{aligned}$$

5.5.5 Property Bridges

Once ConceptBridges are defined, PropertyBridges are defined, and further attached to ConceptBridges. PropertyBridges are responsible for the representation of semantic relations between properties.

In the running example, two PropertyBridges are necessary:

- name2name, semantically relating O1:Individual/name/Literal with O2:Individual/name/Literal. This PropertyBridge must copy instances of the source Path to instances of the target Path. Because both Paths represents ontology attributes, and because a (simple) copy is necessary between instances, the CopyAttribute Service provides the required transformation and is therefore attached to the PropertyBridge:

$$name2name = (\mathcal{W}_1^s, \mathcal{W}_1^t, \{ \}, CopyAttribute, \phi_1^s, \phi_1^t, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{W}_1^s = \{W_{1.1}^s\}$$

$$W_{1.1}^s = O1: Individual / name / Literal$$

$$\mathcal{W}_1^t = \{W_{1.1}^t\}$$

$$W_{1.1}^t = O2: Individual / name / Literal$$

$$\phi_1^s = \{(W_{1.1}^s, \{sourceAttribute\})\}$$

$$\phi_1^t = \{(W_{1.1}^t, \{targetAttribute\})\}$$

- spouseIn2noMariages, which semantically relates O1:Individual/spouseIn/Family with O2:Individual/noMariages/Literal. The target attribute (O2:Individual/noMariages/Literal) represents the number of marriages of O2:Individual, which semantically corresponds to the number of relations O1:Individual has with O1:Family. Therefore, the target attribute should be instantiated with the number of instances of O1:Individual/spouseIn/Family. The CountProperties Service fits this requirements, since it counts the number of instances of a source Path (either AttributePath or RelationPath), and instantiates the target AttributePath with the evaluated number:

$$spouseIn2noMariages := (\mathcal{W}_2^s, \mathcal{W}_2^t, \{ \}, CountProperties, \phi_2^s, \phi_2^t, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{W}_2^s := \{W_{2.1}^s\}$$

$$W_{2.1}^s := O2: Individual / spouseIn / Family$$

$$\mathcal{W}_2^t := \{W_{2.1}^t\}$$

$$W_{2.1}^t := O2: Individual / noMariages / Literal$$

$$\phi_2^s := \{(W_{2.1}^s, \{sourceAttribute\})\}$$

$$\phi_2^t := \{(W_{2.1}^t, \{targetAttribute\})\}$$

PropertyBridges are then \diamond -related with ConceptBridges such that the relation respects the constraint defined in 5.4.7.3, i.e., the root concept of the Paths defined/applied to the PropertyBridge must be the same concepts of the ConceptBridge to which the PropertyBridge is \diamond -related. Because Paths defined in previous PropertyBridges have the O1:Individual and O2:Individual as root concepts, both PropertyBridges must be \diamond -related with Individual2Individual.

$$\diamond_1 = \{(Individual2Individual, \{name2name, spouseIn2noMariages\})\}$$

B_1^P can now be defined as:

$$\mathcal{B}_1^P = \{name2name, spouseIn2noMariages\}$$

5.5.6 Object-Oriented modeling

Notice that the target properties semantically related through the previous PropertyBridges, are not defined only in O1:Individual, but also in both O2:Man and O2:Woman, due to the object-oriented modeling approach allowed in the ontology modeling language and applied in this particular ontology.

If the semantic relation specified in name2name and spouseIn2noMariages are relevant in the scope of Indiv2Man and Indiv2Woman ConceptBridges too, it is necessary to \diamond -relate these two PropertyBridges with these ConceptBridges too:

$$\diamond_1 = \left\{ \left(\text{Indiv2Man}, \{ \text{name2name}, \text{spouseIn2noMariages} \} \right), \left(\text{Indiv2Woman}, \{ \text{name2name}, \text{spouseIn2noMariages} \} \right) \right\}$$

However, this is not the most appropriate specification, especially because SBO provides object-oriented modeling constructs capable to address these scenarios. In particular, SBO provides the following useful constructs:

- Specialization of SemanticBridge into ConceptBridges and PropertyBridges;
- The \prec relation between ConceptBridges;
- The abstract parameter provided by the CopyInstance (primitive) Service, which is the default and unchangeable Service associated with ConceptBridge (5.4.7.4).

In the running example, Indiv2Man and Indiv2Woman are \prec -related with (are sub bridges of) Individual2Individual:

$$\prec_1 = \left\{ \left(\text{Indiv2Man}, \text{Individual2Individual} \right), \left(\text{Indiv2Woman}, \text{Individual2Individual} \right) \right\}$$

Yet, because no O2:Individual instances are expected, but instead instances of either O2:Man or O2:Woman, O2:Individual is considered an abstract concept and therefore the Individual2Individual ConceptBridge should not be executed. Preventing the ConceptBridge to execute is the role of the abstract parameter, provided by the CopyInstance Service. Consequently, the Individual2Individual definition must be modified in order to encompass this requirement:

$$\text{Individual2Individual} = \left(\text{O1: Individual}, \text{O2: Individual}, \mathcal{Q}_{I2I}, \delta_{I2I}^s, \{ \} \right)$$

$$\mathcal{Q}_{I2I} = \{ \text{true} \}$$

$$\delta_{I2I}^s = \{ \text{abstract}(\text{true}) \}$$

As consequence, the extra \diamond -relations defined earlier in this section are not necessary and are even redundant.

5.6 Conclusions

One of the SBO main characteristics is its service-based specification, allowing the description of the ontology mapping system capabilities and its eventual evolution. Table 5.6⁴⁰ summarizes and compares the support provided by SBO in respect to the required support.

Table 5.6 – Semantic relations characteristics supported by the SBO

Dimensions		Characteristics	Required support	SBO support
1. Entity type		Concept to Concept	Yes	Full
		Concept to Property	Yes	Full
		Property to Concept	Yes	Partial
		Property to Property	Yes	Full
		Entity to Instance	Limited	Partial
2. Transformation	2.1 Function	Set of basic functions	Yes	Full
		Combination of functions	Yes	Partial
		Integration of new functions	Yes	Full
	2.2 Directionality	Unidirectional	Yes	Yes
		Manually bidirectional	Limited	No
		Automatically Bidirectional	Limited	No
3. Cardinality		0:1	Yes	n/a
		0:n	n 0:1 bridges	n/a
		1:1	Yes	n/a
		1:n	Yes	n/a
		n:0	Limited	n/a
		n:1	Yes	n/a
		m:n	m:1 and 1:n bridges	n/a
4. Constraint		Not constrained	Yes	Full
		Ontological entities-based	Yes	Yes
		Non-ontological entities-based	Yes	Yes
5. Structural support		Object-oriented	Yes	Yes
		Property-centric	Yes	Yes
		Flow execution control	Yes	Yes

Some of the required support is not directly addressed by SBO itself, but instead it provides the mechanisms that will further permit supporting many of the envisaged characteristics. This is for example the case of the cardinality of the semantic relations that is ultimately defined by the

⁴⁰ “Partial” means that SBO does not directly and fully supports the characteristic, but provides the mechanisms for further support it. “n/a” means that the support of the characteristic does not concern to SBO and is not further addressed. “Full” means that full support is provided for the characteristic according to the envisaged support. “Yes” is stated in contrast to “Full” to reflect the fact that the characteristic is supported but due to its nature it is unclear what full support is.

available Services. In fact, as been argued in 5.2.2, the transformation dimension of the semantic relation is one of the most prevalent, preponderant and orthogonal characteristic of the semantic relation, in what it means, implies and provides to the other characteristics.

Notice that the bidirectionality is not automatically supported by SBO since it has been observed that this characteristic is highly dependent on the Service. As consequence, its support is provided by the Service, which is competent to decide its behavior:

- The Service transformation function is bidirectional (bijective function);
- The Service is not able to perform the inverse relation;
 - It suggests other Service that would perform the inverse relation;
 - No inverse function exists;
 - An inverse function eventually exists but no Service performing that function is known.

Example 5.22 – Characterization of Service according to its inverse Service

The concatenation Service may determine that the Split Service is able to perform the inverse relation of the transformation performed by the concatenation Service.

One of the goals in specifying the Semantic Bridging Ontology has been to minimize new constructs and to maintain and promote existent ones, especially concerning the representation languages of the Semantic Web. This approach promotes the SBO acceptance and understanding by agents and manipulation tools in general, and in the scope of the Semantic Web in particular. [Maedche *et al.*, 2002b; Silva & Rocha, 2003e]. This has been one of the main reasons why SBO has been proposed in the Semantic Web representation languages, such as DAML+OIL and RDFS [SBO].

Notice however, that none of these ontology representation languages is expressive enough to represent all the constraints defined in SBO. SBO has been partially represented in DAML+OIL and RDFS, but its fully representation has been done in the implementation only. This is not a drawback of SBO or of the representation languages, but a simple observation on the expressive power of current ontology representation languages for the Semantic Web. Because no large improvements should be necessary, ontology representation languages would not evolve that much in next years. Instead, a logic and inference layers will be specified above the representation layer that will expand the specification and proofing capabilities as specifically required (Figure 8.1). However, due to the formal specification of SBO, multiple notations, syntaxes and representation mechanisms may be used in the future.

Chapter 6

EXECUTION PROCESS

This chapter describes the work developed during this thesis concerning with the execution phase of the MAFRA - MApping FRAmework. The work described in this chapter has been first introduced in [Silva & Rocha, 2003d; Silva & Rocha, 2004b].

The first section formalizes the execution phase according to the semantic bridging output, and generically describes the proposed execution process based on the relation between ConceptBridges and PropertyBridges. The second section concerns with the details and formalization of the execution process behind the SemanticBridges. The third section presents the extensional specification mechanism, developed to overcome the semantic heterogeneity that requires semantic relations between properties and concepts. The fourth section presents the develop method to constrain the execution of SemanticBridges according to the evaluated target instances. The fifth section presents an overview of this chapter and highlights its fundamental contributions.

6.1 Execution process overview

The execution process described in this chapter relates with the MAFRA Execution module (4.3.4). In this phase, a set of source instances give raise to a set of target instances according to the ontology mapping document \mathcal{M} specified in the semantic bridging phase:

$$\mathbf{T}(\mathcal{M}) \subseteq 2^{I^s} \times 2^{I^t}$$

The execution process is directly and intimately related to the semantic bridging phase and to the semantics of the SBO. Like SBO that distinguishes between ConceptBridge and PropertyBridge, which in turn reflects the ontology model, the execution process distinguishes between:

- The execution of all ConceptBridges, responsible for transforming all referred source instances into target concept instances;
- The execution of all PropertyBridges responsible for creating the properties instances for all created target concept instances.

This process corresponds to the simple flowchart diagram of Figure 6.1:

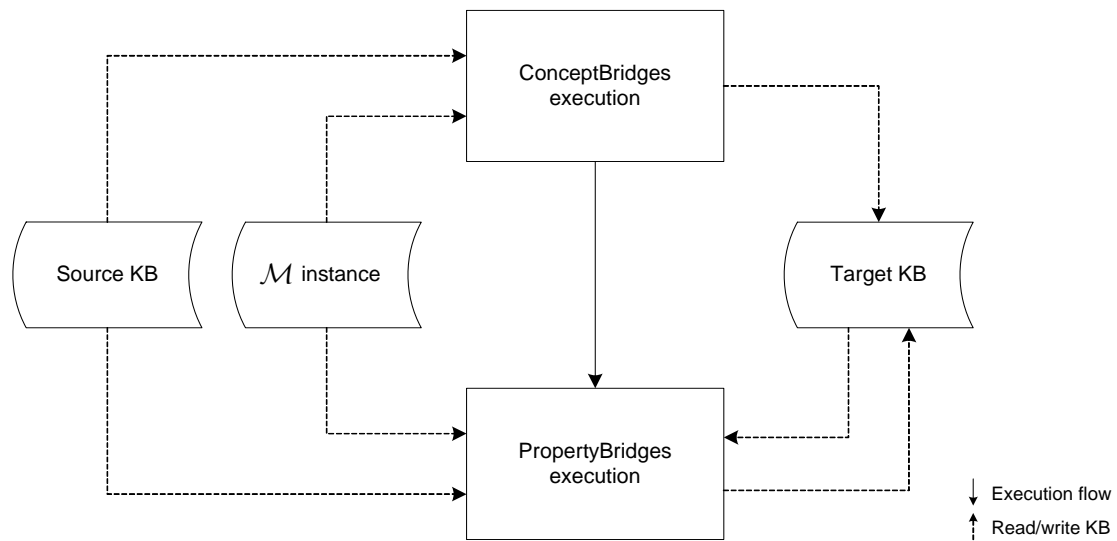


Figure 6.1 –Simple representation of the execution process

The execution process is ultimately dependent on the transformation Services available in the system. As referred in 5.4.3.1, the transformation Service applied in ConceptBridges is made implicit and fixed due to its specificities. Instead, Services applied in PropertyBridges change according to the transformation requirements.

Because concept instances are aggregations of properties instances, and following the approach described in 5.4.3.1, PropertyBridges are executed for and in the scope of every target concept instance.

6.1.1 ConceptBridge execution

The CopyInstance Service implicitly associated with ConceptBridges is formally defined as follows:

$$S_{CopyInstance} = \left(\text{"pt.ipp.isep.gecad.mafra.services.transformations.CopyInstance"}, \right)$$

$$Args_{CopyInstance}^s = \{sourceConcept\}$$

$$Args_{CopyInstance}^t = \{targetConcept\}$$

$$\alpha_{CopyInstance}^s = \{(sourceConcept, concept)\}$$

$$\alpha_{CopyInstance}^t = \{(targetConcept, concept)\}$$

While during semantic bridging phase the CopyInstance Service means that the source ontology concept is semantically related with the target ontology concept, at execution phase it means that every instance of source ontology concept will give raise to an instance of the target ontology concept⁴¹.

Every newly created target concept instance is characterized by:

1. The concept it represents, which defines and constrains its semantics and properties;
2. Its unique identification. For every newly created target instance, a (new) identity is associated with it. Three main reasons justify this decision:
 - The target concept instance is different from the source instance, once they are defined according to two distinct ontologies;
 - The target concept instance is defined in a new repository;
 - Many target concept instances may arise from the same source concept instance.

Because the properties of target concept instances are created according to the properties of the source concept instances, it is necessary to relate every target concept instance with the source concept instance from which it has been generated. Thus, a map relating the every pair of source and target concept instances is created for every target concept instance created. This information is named Transformation Information (*TI*) and takes the form of tuple in the form of:

$$TI := (source_concept_instance, target_concept_instance)$$

However, this information must be extended in order to support some other requirements. In particular, notice that:

- The properties of the target instance are created by the PropertyBridges \diamond -related with the ConceptBridge responsible for the target instance creation;

⁴¹ Later, this specification will be expanded to accommodate some other transformation requirements.

- The set of PropertyBridges \diamond -related with ConceptBridges varies from ConceptBridge to ConceptBridge, which means that different PropertyBridges will be executed in the scope of each target concept instance.

It is therefore necessary to keep track of the ConceptBridge responsible for the transformation of the target instance. This implies the expansion of the transformation information tuple to:

$$TI := (source_concept_instance, target_concept_instance, concept_bridge)$$

The set of transformation information tuples is named transformation information table, and is represented by TI^2 .

Example 6.1 – Execution process of ConceptBridges

Consider the ontology mapping scenario of Figure 6.2, previously introduced in 5.5.1. The O1:Family concept has been expanded with “marriage” and “divorce” properties, relating O1:Family to O1:Event. Additionally, O1:Individual concept is now related with O1:Event through the “birth” relation.

This very simple ontology mapping scenario permits the description and analysis of the mechanisms behind the ConceptBridges execution process. Three ConceptBridges are defined between source and target ontology: Indiv2Indiv, Indiv2Man and Indiv2Woman. Indiv2Man and Indiv2Woman are sub-bridges of Individual2Individual and mutually exclusive.

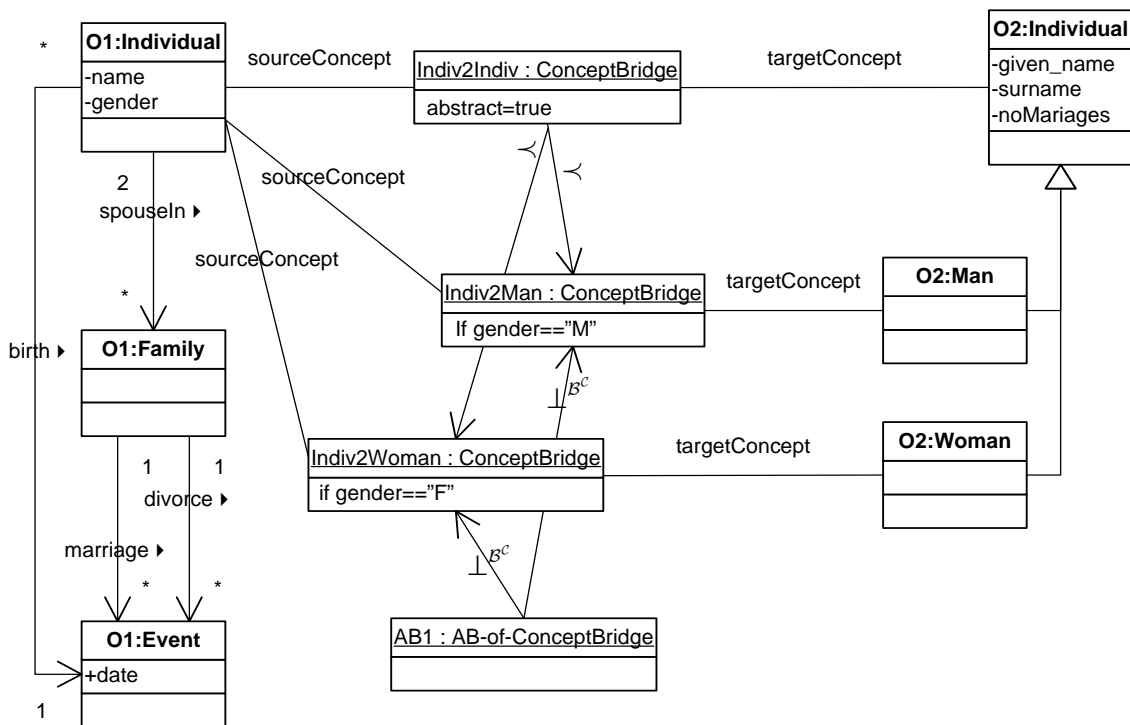


Figure 6.2 – UML-like representation of ontology mapping scenario

Notice that the notation used to specify the ConditionExpressions (e.g. if gender=="F") is a simplification of UML and SBO.

Also, consider the following excerpt of the source knowledge base:

$$\begin{aligned} \mathcal{I}_{O_1} &= \{i_{1.1}, i_{1.2}, i_{1.3}, i_{1.4}, i_{1.5}, f_{1.1}, f_{1.2}, f_{1.3}, e_{1.1}, e_{1.2}, e_{1.3}, e_{1.4}\} \\ \text{inst}\mathcal{C}_{O_1} &= \left\{ \begin{array}{l} \text{Individual}(i_{1.1}), \text{Individual}(i_{1.2}), \text{Individual}(i_{1.3}), \text{Individual}(i_{1.4}), \\ \text{Individual}(i_{1.5}), \text{Family}(f_{1.1}), \text{Family}(f_{1.2}), \text{Family}(f_{1.3}), \\ \text{Event}(e_{1.1}), \text{Event}(e_{1.2}), \text{Event}(e_{1.3}), \text{Event}(e_{1.4}) \end{array} \right\} \\ \text{inst}\mathcal{P}_{O_1} &= \left\{ \begin{array}{l} \text{gender}(i_{1.1}, "M"), \text{gender}(i_{1.2}, "F"), \text{gender}(i_{1.3}, "F"), \\ \text{gender}(i_{1.4}, "M"), \text{gender}(i_{1.5}, "F") \end{array} \right\} \end{aligned}$$

The execution process upon this scenario is rather straightforward because each source concept is semantically bridged to a unique target concept. In fact, notice that *Indiv2Indiv* *ConceptBridge* is stated abstract and therefore no *O2:Individual* instances will be created. *Indiv2Man* and *Indiv2Woman* *ConceptBridges* are mutually exclusive, which means that from each *O1:Individual* instance, a unique target concept instance (either *O2:Man* or *O2:Woman*) will be created. Accordingly, the transformation information respecting this scenario corresponds to the following set:

$$TI_{O_1-O_2}^2 = \left\{ \begin{array}{l} (i_{1.1}, i_{2.1}, \text{Indiv2Man}), (i_{1.2}, i_{2.2}, \text{Indiv2Woman}), (i_{1.3}, i_{2.3}, \text{Indiv2Woman}), \\ (i_{1.4}, i_{2.4}, \text{Indiv2Man}), (i_{1.5}, i_{2.5}, \text{Indiv2Woman}) \end{array} \right\}$$

Previous information corresponds to the following table-based representation:

Table 6.1 – Table-based representation of the Transformation Information Table ($TI_{O_1-O_2}^2$)

Source Concept Instance ID	Target Concept Instance ID	ConceptBridge
$i_{1.1}$	$i_{2.1}$	Indiv2Man
$i_{1.2}$	$i_{2.2}$	Indiv2Woman
$i_{1.3}$	$i_{2.3}$	Indiv2Woman
$i_{1.4}$	$i_{2.4}$	Indiv2Man
$i_{1.5}$	$i_{2.5}$	Indiv2Woman

Accordingly, the target knowledge base will contain the following instances:

$$\begin{aligned} \mathcal{I}_{O_2} &= \{i_{2.1}, i_{2.2}, i_{2.3}, i_{2.4}, i_{2.5}\} \\ \text{inst}\mathcal{C}_{O_2} &= \{\text{Man}(i_{2.1}), \text{Woman}(i_{2.2}), \text{Woman}(i_{2.3}), \text{Man}(i_{2.4}), \text{Woman}(i_{2.5})\} \end{aligned}$$

6.1.2 PropertyBridge execution

PropertyBridges run in the scope of every target concept instance. Because the *ConceptBridge* responsible for the target concept instance creation is known through TI^2 , it is possible to enumerate all *PropertyBridges* that should be executed for each target concept instance.

Moreover, due to the \prec relation between *ConceptBridges*, the sub bridges inherit the *PropertyBridges* from their super bridges, as described in 5.4.7.4 and exemplified in 5.5.6.

Each *PropertyBridge* is executed, and its outcome associated with the target concept instance in which the *PropertyBridge* runs for.

Example 6.2 – Execution process of PropertyBridges

Consider the mapping scenario of section 5.5 where the name2names and spouseIn2noMarriages PropertyBridges are defined and \diamond -related with Indiv2Indiv ConceptBridge. Figure 6.3 illustrates this scenario using the UML notation.

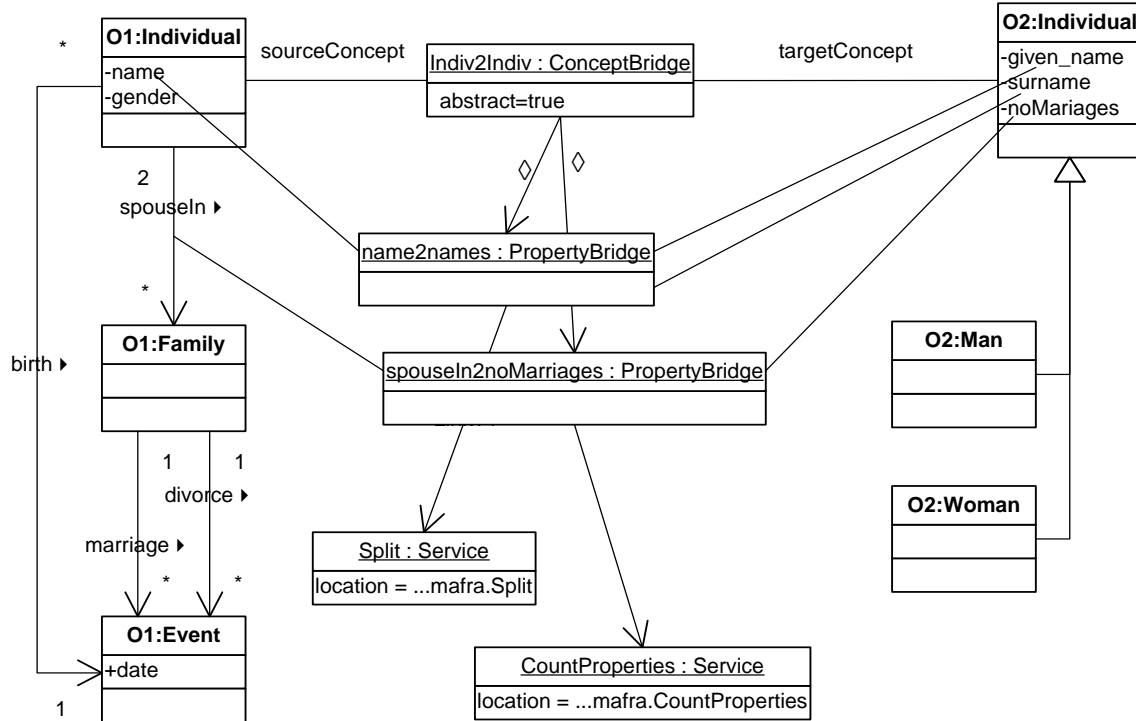


Figure 6.3 – PropertyBridges representation in UML

Also, consider the following knowledge base that extends the previously presented knowledge base, by defining some Literals and property instances:

$$\begin{aligned}
 \mathcal{I}_{O_1} &= \{i_{1.1}, i_{1.2}, i_{1.3}, i_{1.4}, i_{1.5}, f_{1.1}, f_{1.2}, f_{1.3}, e_{1.1}, e_{1.2}, e_{1.3}, e_{1.4}\} \\
 instC_{O_1} &= \left\{ \begin{array}{l} Individual(i_{1.1}), Individual(i_{1.2}), Individual(i_{1.3}), Individual(i_{1.4}), \\ Individual(i_{1.5}), Family(f_{1.1}), Family(f_{1.2}), Family(f_{1.3}), \\ Event(e_{1.1}), Event(e_{1.2}), Event(e_{1.3}), Event(e_{1.4}), Event(e_{1.5}) \end{array} \right\} \\
 instP_{O_1} &= \left\{ \begin{array}{l} gender(i_{1.1}, "M"), gender(i_{1.2}, "F"), gender(i_{1.3}, "F"), gender(i_{1.4}, "M"), \\ gender(i_{1.5}, "F"), birth(i_{1.1}, e_{1.5}), date(e_{1.5}, 1769), \\ name(i_{1.1}, "Napoleon Bonapart"), name(i_{1.2}, "Joséphine de Tasher"), \\ name(i_{1.3}, "Marie – Louise de Austria"), name(i_{1.4}, "William Clinton"), \\ name(i_{1.5}, "Hillary Rodham"), \\ spouseIn(i_{1.1}, f_{1.1}), spouseIn(i_{1.1}, f_{1.2}), spouseIn(i_{1.2}, f_{1.1}), \\ spouseIn(i_{1.3}, f_{1.2}), spouseIn(i_{1.4}, f_{1.3}), spouseIn(i_{1.5}, f_{1.3}), \\ marriage(f_{1.1}, e_{1.1}), divorce(f_{1.1}, e_{1.2}), \\ marriage(f_{1.2}, e_{1.3}), marriage(f_{1.3}, e_{1.4}), \\ date(e_{1.1}, 1796), date(e_{1.2}, 1810), date(e_{1.3}, 1810), date(e_{1.4}, 1975) \end{array} \right\}
 \end{aligned}$$

Previous knowledge based corresponds to the following table-based representation (Table 6.2):

Table 6.2 - Table-based representation of knowledge base

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family	Individual/ birth/ Event
$i_{1.1}$	"Napoleon Bonapart"	"M"	$f_{1.1}$	$e_{1.5}$
$i_{1.1}$	"Napoleon Bonapart"	"M"	$f_{1.2}$	$e_{1.5}$
$i_{1.2}$	"Joséphine de Tasher"	"F"	$f_{1.1}$	
$i_{1.3}$	"Marie-Louise de Austria"	"F"	$f_{1.2}$	
$i_{1.4}$	"William Clinton"	"M"	$f_{1.3}$	
$i_{1.5}$	"Hillary Rodham"	"F"	$f_{1.3}$	

Family/ ID/ Literal	Family/ marriage/ Event	Family/ divorce/ Event
$f_{1.1}$	$e_{1.1}$	$e_{1.2}$
$f_{1.2}$	$e_{1.3}$	
$f_{1.3}$	$e_{1.4}$	

Event/ ID/ Literal	Event/ date/ Literal
$e_{1.1}$	1796
$e_{1.2}$	1810
$e_{1.3}$	1810
$e_{1.4}$	1975
$e_{1.5}$	1769

PropertyBridges execution will run for every target concept instances. Considering the previously presented TI^2 , $i_{1.1}$ and $i_{1.4}$ instances have been created by the Indiv2Man ConceptBridge. No PropertyBridge has been directly \diamond -related with Indiv2Man. Yet, Indiv2Man is sub bridge of Indiv2Indiv thus inheriting its \diamond -related PropertyBridges: name2names and spouseIn2noMarriages. Equivalent situation occurs for the $i_{1.2}$, $i_{1.3}$ and $i_{1.5}$ instances through Indiv2Woman ConceptBridge. The target knowledge base becomes therefore:

$$\begin{aligned} \mathcal{I}_{O_2} &= \{i_{2.1}, i_{2.2}, i_{2.3}, i_{2.4}, i_{2.5}\} \\ instC_{O_2} &= \{Man(i_{2.1}), Woman(i_{2.2}), Woman(i_{2.3}), Man(i_{2.4}), Woman(i_{2.5})\} \\ instP_{O_2} &= \left\{ \begin{array}{l} given_name(i_{2.1}, "Napoleon"), given_name(i_{2.2}, "Joséphine"), \\ given_name(i_{2.3}, "Marie - Louise"), given_name(i_{2.4}, "William"), \\ given_name(i_{2.5}, "Hillary"), surname(i_{2.1}, "Bonapart"), \\ surname(i_{2.2}, "de Tasher"), surname(i_{2.3}, "de Austria"), \\ surname(i_{2.4}, "Clinton"), surname(i_{2.5}, "Rodham"), \\ noMarriages(i_{2.1}, "2"), noMarriages(i_{2.2}, "1"), noMarriages(i_{2.3}, "1"), \\ noMarriages(i_{2.4}, "1"), noMarriages(i_{2.5}, "1") \end{array} \right\} \end{aligned}$$

Despite no internal process details have been presented so far, this generic description shows that the essence of the ontology mapping execution phase process is very clear and maintains the essence of SBO philosophy.

6.2 Internal process

During the semantic bridging phase, Paths are applied in establishing the ontology entities semantically related in the SemanticBridge, which are applied in the execution phase both in accessing the source knowledge base and creating new target knowledge base instances. Paths are therefore the basic elements for reading and writing knowledge bases.

The execution process is divided into three phases:

1. Querying the source knowledge base according to the source Paths defined in the SemanticBridges, which result in particular views upon the source concept instances;
2. Filter the knowledge base query according to ConditionExpressions, constraining the view of the properties values of the source concept instances, previously evaluated;
3. Create the target knowledge base instances according to the filtered properties instances evaluated in previous phase.

Next sections will describe each of these three phases.

6.2.1 Querying the source knowledge base

In order to describe the developed approach, the relational data model [Codd, 1970] and the corresponding relational algebra will be used. Annex 1 provides simple insights on the fundamentals of the relational data model as used during this thesis.

Paths are treated equally independently if they are applied in SemanticBridges as arguments for Services or in ConditionExpressions. Notice that only the source Paths defined in the SemanticBridge are considered in this phase. Each source Path corresponds to a specific perspective of the source concept instance according to the properties values.

As referred in 6.1, SemanticBridges are always executed in the scope of a source concept instance (*SCI*), which corresponds to all property values of the instance, i.e. the Cartesian product of all property values. This corresponds to a table whose attributes correspond to the concept properties and the attributes values are the properties values. The KB querying always occurs upon a source concept instance (*SCI*). Table 6.3 is the table-based representation of the $i_{1,1}$ *SCI*.

Table 6.3 - Table-based representation of the $i_{1,1}$ source concept instance

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family	Individual/ birth/ Event
$i_{1,1}$	“Napoleon Bonapart”	“M”	$f_{1,1}$	$e_{1,5}$
$i_{1,1}$	“Napoleon Bonapart”	“M”	$f_{1,2}$	$e_{1,5}$

A single step Path query results in a certain perspective of the *SCI*. In fact, because the property specified in the unique Step is in *SCI*, query the KB through a single step Path corresponds algebraically to project *SCI* according to the first and unique Step of the Path.

These relational operations are denoted by the λ operator:

$$\lambda_{Path}T := \pi_{Path[1]}T$$

where:

- T is any valid set of source concept instances (a table) or *SCI*;
- $Path[1]$ is the first Step of the Path, which in this case is unique.

Notice that the query operation is based on the Path and not on the Step of the Path. In this particular case, because the Path is composed by one single Step, the result is the same.

Example 6.3 – Querying a single-Step Path

The $\lambda_{Individual/spouseIn/Family}i_{1,1}$ operation upon the knowledge base presented in section 6.1 results in the set $\{f_{1,1}, f_{1,2}\}$, since source instance $i_{1,1}$ has two instances of spouseIn property. This corresponds in natural language to state that “Napoleon Bonapart married twice”. Table 6.4 is the table-based representation of previous operation:

Table 6.4 – Result of the single-Step Path query

Individual/spouseIn/Family
$f_{1,1}$
$f_{1,2}$

Querying the KB through multi-step Paths is more complicated. Pragmatically, it corresponds to enumerate all values of the last Step in the Path, such there is a relationship from the source concept instance through every Step of the Path. Algebraically, it corresponds to consecutively “left join” the concepts defined in the Path. Adopting an experimental approach, querying a *SCI* through a Path with n Steps corresponds to:

$$\begin{aligned}
 T_1 &= SCI \mid_{*(S_1/P_1/O_1),(O_1/ID/Literal)} O_1 \\
 T_2 &= T_1 \mid_{*(S_2/P_2/O_2),(O_2/ID/Literal)} O_2 \\
 &\dots \\
 T_{n-1} &= T_{n-2} \mid_{*(S_{n-1}/P_{n-1}/O_{n-1}),(O_{n-1}/ID/Literal)} O_{n-1} \\
 T_n &= \pi_{Path} \left(\rho_{Path/Step_n} T_{n-1} \right)
 \end{aligned}$$

where:

- S_i and P_i are respectively the subject and predicate of the Step of order i in the Path;
- O_i corresponds to:
 - The object of the Step of order i in the Path, when used as a parameter to the operator;

- To the instances of concept O_i when used in place of a table.

Previous operations can be systematized into the recursive λ operator as follows:

$$\begin{aligned}
 [1] \quad \lambda_{Path} T &:= \pi_{Path} \left(\rho_{Path / Path[length(Path)]} \left(\eta_{Path[1], Path-Path[1]} T \right) \right) \\
 [2] \quad \eta_{[S/P/O], []} T &:= T \\
 [3] \quad \eta_{Step, Path} T &:= \eta_{Path[1], Path-Path[1]} \left(\omega_{Step} T \right) \\
 [4] \quad \omega_{S/P/O} T &:= T \mid_{*(S/P/O), (O/ID/Literal)} O
 \end{aligned}$$

This operator comprehends four distinct phases, inversely described for better explanation:

- [4] For each Step in the Path, a left join operation is performed between the table resulting from previous Steps (T) and the table representing the instances of object concept (O). The join attributes are the current Step ($S/P/O$) and the ID attribute of the object concept table;
- [3] Previous operation will be executed for each Step in the Path except for the last Step (line [2]). The table resulting from the query of each Step will be applied as the input table for the next Step;
- [2] For the last Step of the Path no query operation is required since all the attributes of the subject are already presented in the table;
- [1] Because the query corresponds to a table with one column named after the Path, it is necessary to project the last column into a 1-column table and rename it to Path.

Example 6.4 – Left-join operation of a forward Step

Figure 6.4 depicts this situation for the $\omega_{Individual / spouseIn / Family} T$ operation. T is the left table in the figure, which corresponds to the $i_{1,1}$ instance of previous KB⁴².

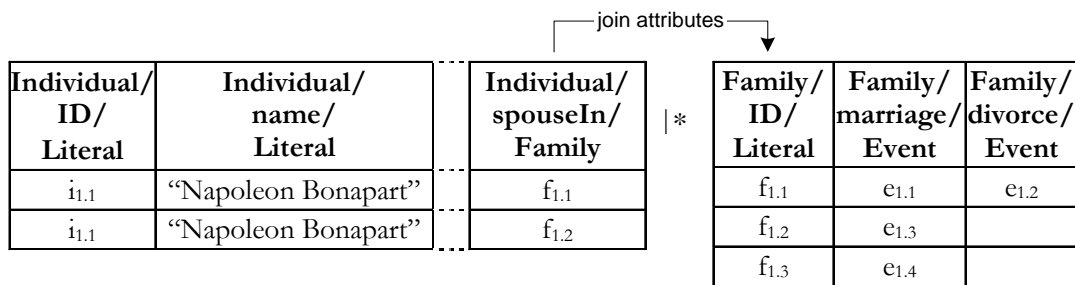


Figure 6.4 – Schematic representation of the left-join attributes in a forward Step

⁴² The O1:Individual/gender/Literal and O1:Individual/birth/Event properties are not represented in the table in order to maintain the scenario as simple as possible. The same approach is followed in the next examples.

The result of previous left-join operation is presented in Table 6.5:

Table 6.5 – Query result of $\omega_{\text{Individual/spouseIn/Family}}^{i_{1.1}}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ spouseIn/ Family	Family/ ID/ Literal	Family/ marriage/ Event	Family/ divorce/ Event
$i_{1.1}$	“Napoleon Bonapart”	$f_{1.1}$	$f_{1.1}$	$e_{1.1}$	$e_{1.2}$
$i_{1.1}$	“Napoleon Bonapart”	$f_{1.2}$	$f_{1.2}$	$e_{1.3}$	

However, this query operator works only for forward Paths, i.e. Paths composed only by Steps whose direction attribute is set to “forward”. Yet, as defended in 5.2.2.2 and further supported in SBO, backward Steps (backward Paths) are a fundamental construct in ontology mapping, that should therefore reflect distinct but coherent queries.

Because directionality is defined in the scope of each Step, and because line [2] and line [4] are the only lines that process Steps, the changes required in λ operator are concentrated in these lines.

Backward Step (S\P\O) determines all instances of concept O that relate to instances of S through property P.

Example 6.5 – Left-joint operation of a backward Step

Consider the Individual/spouseIn/Family\spouseIn\Individual Path that represents the Individuals that are married in the same Families.

Because the first Step of the Path is the same as the Path applied in previous example, this example concerns with the $\omega_{\text{Family\spouseIn\Individual}}^T$ operation only. Thus, input table

T corresponds to the table resulting from previous example (Table 6.5). Figure 6.5 illustrates the left join operation between T (partially presented) and the instances of the subject concept defined in next Step.

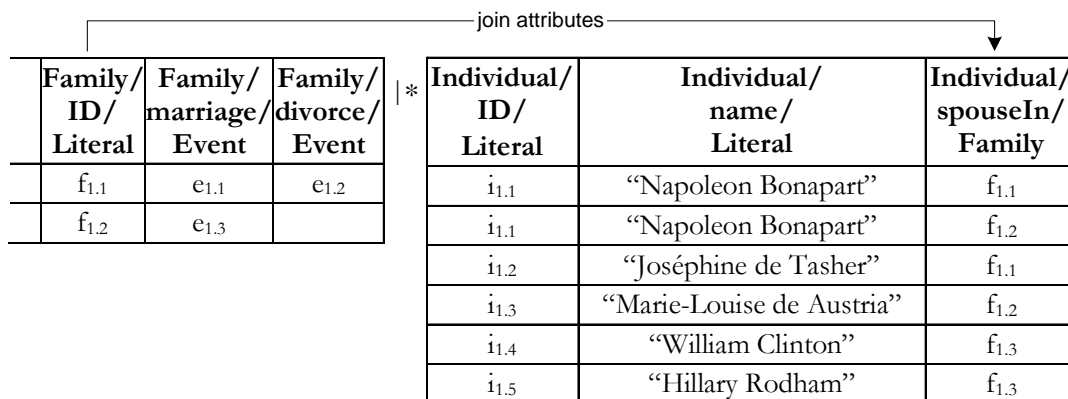


Figure 6.5 – Schematic representation of left-join attributes in a backward Step

The result of previous left join operation is presented in Table 6.6:

Table 6.6 – Query result of $\omega_{Family\setminus spouseIn\setminus Individual}T$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ spouseIn/ Family	Family/ ID/ Literal	Family/ marriage/ Event
i _{1,1}	“Napoleon Bonapart”	f _{1,1}	f _{1,1}	e _{1,1}
i _{1,1}	“Napoleon Bonapart”	f _{1,1}	f _{1,1}	e _{1,1}
i _{1,1}	“Napoleon Bonapart”	f _{1,2}	f _{1,2}	e _{1,3}
i _{1,1}	“Napoleon Bonapart”	f _{1,2}	f _{1,2}	e _{1,3}

Family/ divorce/ Event	Individual\ ID\ Literal	Individual\ name\ Literal	Individual\ spouseIn\ Family
e _{1,2}	i _{1,1}	“Napoleon Bonapart”	f _{1,1}
e _{1,2}	i _{1,2}	“Joséphine de Tasher”	f _{1,1}
	i _{1,1}	“Napoleon Bonapart”	f _{1,2}
	i _{1,3}	“Marie-Louise de Austria”	f _{1,2}

In order to support backward Paths, the λ operator should be extended to:

$$[1] \quad \lambda_{Path}T := \pi_{Path} \left(\rho_{Path / Path[length(Path)]} \left(\eta_{Path[1], Path-Path[1]}T \right) \right)$$

$$[2.1] \quad \eta_{[S/P/O], []}T := T$$

$$[2.2] \quad \eta_{[S \setminus P \setminus O], []}T := \rho_{(S \setminus P \setminus O), (O / P / S)} \left(T \mid_{*(S / ID / Literal), (O / P / S)} O \right)$$

$$[3] \quad \eta_{Step, Path}T := \eta_{Path[1], Path-Path[1]} \left(\omega_{Step}T \right)$$

$$[4.1] \quad \omega_{S/P/O}T := T \mid_{*(S/P/O), (O/ID/Literal)} O$$

$$[4.2] \quad \omega_{S \setminus P \setminus O}T := T \mid_{*(S/ID/Literal), (O/P/S)} O$$

From the previous specification, two lines have been extended:

- Line [2] gave rise to line [2.1] and line [2.2]. While line [2.1] corresponds to line [2] of previous definition, line [2.2] has been added to addresses backward Steps. Unlike last forward Steps of the Path, whose attributes are present in the input table (T) and therefore no operation is required, in backward Steps a left join operation is necessary between T and the table that represents the subject of the backward Step (O). The rename operation is necessary because of the rename operation in line [1], which expects a Step as it is specified in the Path (backward Step) an not as it is represented in the table (forward Step);
- Line [4] gave rise to line [4.1] and [4.2]. While line [4.1] is identical to line [4] of previous specification, line [4.2] processes the backward Step as in line [2.2] (except the rename operation that is not necessary).

This version of the λ operator provides the functionalities for querying both single-Step and multi-Step Paths and therefore it fully substitutes the previous specification. Still, it will be slightly modified later (6.2.1.2) to address a special requirement.

This operation is executed for every Paths of the SemanticBridge and the outcome is a table for every Path (T_{Path}) whose unique column is also named after the Path.

Example 6.6 – Querying through a multi-Step backward Path

From the knowledge base presented in 6.1, the $\lambda_{\text{Individual/spouseIn/Family}\backslash\text{spouseIn}\backslash\text{Individual}}^{i_{1.1}}$ results in Table 6.7:

Table 6.7 – Result of the query through a multi-Step backward Path

Individual/spouseIn/Family\spouseIn\Individual
$i_{1.1}$
$i_{1.2}$
$i_{1.1}$
$i_{1.3}$

This corresponds to enumerate all the Individuals involved in the marriages in which “Napoleon Bonapart” is involved, including himself.

This phase proceeds by combining the n 1-column tables into one n -column table. The simplest method to combine tables is through the Cartesian product:

$$T := T_{Path_1} \times T_{Path_2} \times \dots \times T_{Path_n}$$

However, in some cases, Paths in the same SemanticBridge have sub-Paths in common, which causes referential constraints between Path queries. In such cases, the Cartesian product is meaningless and derives semantic errors.

Example 6.7 – Querying multiple Paths without addressing referential constraints

Consider that the following Paths have been defined in the same SemanticBridge:

- Individual/spouseIn/Family
- Individual/spouseIn/Family/marriage/Event/date/Literal

The Cartesian product of these Paths, upon the $i_{1.1}$ *SCI* of the knowledge base presented in Example 6.2, is represented in Table 6.8:

Table 6.8 – Cartesian product of the two previous tables

Individual/spouseIn/Family	Individual/spouseIn/Family/marriage/Event/date/Literal
$f_{1.1}$	1796
$f_{1.1}$	1810
$f_{1.2}$	1796
$f_{1.2}$	1810

As noticeable, some of previous relations are semantically incorrect. For example, Family $f_{1.2}$ has no relation with the date of 1796.

Therefore, the combination of tables based on Cartesian product is admissible only between Paths without common parts. In case Paths share a common initial sub-Part, a slightly different process is necessary in order to comply with referential constraints.

To describe and systematize the necessary combination process, an experimental approach is followed. Consider the following Paths:

- Individual/spouseIn/Family
- Individual/spouseIn/Family/marriage/Event/date/Literal
- Individual/spouseIn/Family/divorce/Event/date/Literal

Notice that both have the first Step in common (i.e. Individual/spouseIn/Family), as required by constraints in 5.4.7.3. The result of the query based on this Step constrains the query of the next Steps. Therefore, a “left join” operation is required between the table resulting from the common sub-Path query, and the tables resulting from the query for the rest of the Paths. The results of the rest of the Paths are left joined through the common Path attribute (referential constraint). The necessary operations are as follows:

$$Path_0 = \text{Individual} / \text{spouseIn} / \text{Family}$$

$$Path_1 = \text{Individual} / \text{spouseIn} / \text{Family} / \text{marriage} / \text{Event} / \text{date} / \text{Literal}$$

$$Path_2 = \text{Individual} / \text{spouseIn} / \text{Family} / \text{divorce} / \text{Event} / \text{date} / \text{Literal}$$

$$S_0 = SCI \mid_{(Individual/spouseIn/Family),(Family/ID/Literal)}^* Family$$

$$S_1 = \rho_{Path_1/(Event/date/Literal)} \left(\pi_{Path_0,Event/date/Literal} \left(S_0 \mid_{(Family/marriage/Event),(Event/ID/Literal)}^* Event \right) \right)$$

$$S_2 = \rho_{Path_2/(Event/date/Literal)} \left(\pi_{Path_0,Event/date/Literal} \left(S_0 \mid_{(Family/divorce/Event),(Event/ID/Literal)}^* Event \right) \right)$$

$$T_{final} = S_1 \mid_{(Path_0),(Path_0)}^* S_2$$

Executing previous operations for the $i_{1.1}$ SCI , T_{Final} would correctly result in Table 6.9:

Table 6.9 – Multiple Paths with common sub-Path querying

Individual/spouseIn/ Family	Individual/spouseIn/Family/ marriage/Event/ date/Literal	Individual/spouseIn/Family/ divorce/Event/ date/Literal
$f_{1.1}$	1796	1810
$f_{1.2}$	1810	

The previous experimental process can be systematized into the following steps:

- Query the KB according to common sub Path (S_0);
- The resulting table is “left joined” with the query of distinct branches of the Path (S_1 and S_2);

- The query of every distinct branch is combined with the others through the left join operation, whose join attributes is the common sub-Path.

Because the described process depends on the characterization of Paths according to their common/distinct parts, it is necessary to represent Paths accordingly.

6.2.1.1 Tree-based representation of Paths

The specified solution adopts a tree-based representation of Paths, in which the tree nodes represent the common sub-Paths and the tree branches represent the distinct sub-Paths.

Two or more Paths are considered distinct if their first Step is different from each other. In such case they are represented in different branches of the tree. Instead, if they have the first Step in common they (partially) belong to the same branch of the tree. In case two or more Paths have distinct first Steps but share certain sub-Path after the first Step, these Paths are still considered distinct.

Example 6.8 – Paths with common sub-Paths

Consider the UML representation of the ontology presented in Figure 6.6 and the following Paths upon the ontology, all applied in the same SemanticBridge:

1. *O1:Individual / spouseIn / Family*
2. *O1:Individual / spouseIn / Family / marriage / Event / date / Literal*
3. *O1:Individual / spouseIn / Family / divorce / Event / date / Literal*
4. *O1:Individual / birth / Event / date / Literal*

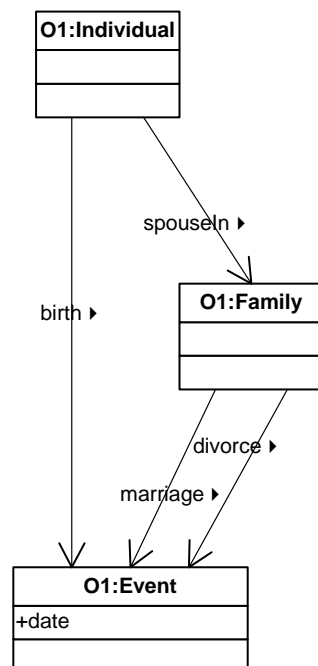


Figure 6.6 – UML representation of ontology

Observe that first, second and third Paths have the same first Step (i.e. Individual/spouseIn/Family). Therefore they belong to the same branch of the tree. Instead, fourth Path belongs to a different branch even if they all share the Event/date/Literal sub-Path.

The common sub-Path of a branch of the tree is the longest sub-Path that is common to all Paths in the branch. Hence, the common sub-Path of a branch can be longer than one-Step. The distinct sub-Paths in a branch form sub-branches, which in turn are treated as any other branch, with common sub-Path and distinct sub-Paths. Accordingly, the adopted tree-based representation of Paths is defined according to the following recursive structure:

$BRANCHES ::= BRANCH *$

$BRANCH ::= \langle PATH, BRANCHES \rangle$

$PATH$ is a Path specified in the SemanticBridge, or a sub-Path common to all (sub-) $BRANCHES$ of the tuple.

Example 6.9 – Tree-based representation of Paths

The Paths of Example 6.8 are represented in the following tree-based structure. Because this example will be used later, every component of the Paths is explicitly named.

$$\begin{aligned}
 Paths_{01} &= \{ Paths_{01-1}, Paths_{01-2} \} \\
 Paths_{01-1} &= \langle CP_{01-1}, Paths_{01-1D} \rangle \\
 Paths_{01-1D} &= \{ Paths_{01-1-1}, Paths_{01-1-2} \} \\
 Paths_{01-1-1} &= \langle CP_{01-1-1}, Paths_{01-1-1D} \rangle \\
 Paths_{01-1-1D} &= \{ \} \\
 Paths_{01-1-2} &= \langle CP_{01-1-2}, Paths_{01-1-2D} \rangle \\
 Paths_{01-1-2D} &= \{ \} \\
 Paths_{01-2} &= \langle CP_{01-2}, \{ \} \rangle \\
 CP_{01-1} &= Individual / spouseIn / Family \\
 CP_{01-1-1} &= Family / marriage / Event / date / Literal \\
 CP_{01-1-2} &= Family / divorce / Event / date / Literal \\
 CP_{01-2} &= Individual / birth / Event / date / Literal
 \end{aligned}$$

To access and manipulate this information structure, three constructs have been specified:

- $Paths[n]$ provides access to the n th branch defined in the $Paths$ tree-based representation of Paths.

Example 6.10 – Array-like access to tree-based represented Paths

Considering the previously defined tree-based representation of Paths, then $Paths_{01}[1] == Paths_{01-1}$

Multiple levels can be specified to access nested branches using the array-like nomenclature $Paths[n][m]..[x]$.

Example 6.11 – Multi-dimension array-like access to tree-based represented Paths

Considering again previous Paths, then $Paths_{O_1}[1][2] == Paths_{O_1-1-2}$;

- $distinctBranches(Branch)$ returns the set of all distinct branches of a branch.

Example 6.12 – Retrieving distinct Branches of tree-based represented Paths

The set of distinct branches for the “Individual/spouse/Family” branch of previous Paths (i.e. $Paths_{O_1}[1]$), is defined by $distinctBranches(Paths_{O_1}[1]) == Paths_{O_1-ID}$;

- $commonPath(Branch)$ returns the Path that is common to all distinct branches of $Branch$.

Example 6.13 - Retrieving the common Path of a tree-based represented Path

The Path common to all Paths belonging to the first branch of $Paths_{O_1}$ is defined by: $commonPath(Paths_{O_1}[1]) == Individual / spouseIn / Family$.

6.2.1.2 Tree-based query

Querying the knowledge base according to previous tree-based representation of Paths comprehends the following operations:

- Query KB according to each distinct branch and combine them either through:
 - The Cartesian product if they have no attributes in common;
 - The left join if they share a sub-Path;
- Process every distinct branch as follows:
 - Query the KB according to the branch common sub-Path based on previous table;
 - The resulting table serves as input for querying KB according to every distinct branch of the sub-Path.

This corresponds to the following φ operator:

$$[1] \quad \varphi_{\{ \}} T := T$$

$$[2] \quad \varphi_{Branches} T := \left(\varphi_{Branches-Branches[1]} T \right) \times \left(\tau_{Branches[1], commonPath(Branches[1])[1]} \left(\pi_{commonPath(Branches[1])[1]} T \right) \right)$$

$$[3] \quad \psi_{\{ \}, PathP} T := T$$

$$[4] \quad \psi_{Branches, PathP} T := \left(\psi_{Branches-Branches[1], PathP} T \right) |_{*(PathP), (PathP)}^* \left(\tau_{Branches[1], PathP} T \right)$$

$$[5] \quad \tau_{Branch, PathP} T := \psi_{disjointBranches(Branch), commonPath(Branch)} \left(\nu_{commonPath(Branch), PathP} T \right)$$

$$[6] \quad \nu_{Path, PathP} T := T |_{*(PathP), (PathP)}^* \left(\lambda_{Path, PathP} T \right)$$

These lines correspond to the following processes:

- [1] This line corresponds to the stop condition when dispatching for processing Paths that have only the root concept in common (e.g. *Individual / spouseIn / Family* and *Individual / birth / Event / date / Literal*);
- [2] This line dispatches every of those distinct Paths for processing. The input table is projected (π) in order to maintain only the attribute corresponding to the first Step of the Path (e.g. *Individual / birth / Event*). This attribute will serve as join attribute in next lines. Resulting tables are combined through the Cartesian product because they have no attribute in common;
- [3] This line corresponds to the stop condition when dispatching distinct branches of Paths that have a common sub-Path;
- [4] This line dispatches every distinct branch of the Path for processing. The resulting tables are left joined through the attribute corresponding to the common sub-Path of the branches;
- [5] This line queries the KB according to every branch. The first part of the line dispatches the distinct sub-branches of the branch for processing. The second part of the line dispatches the common Path of the branch for query. The result will serve as input for the first part of the line;
- [6] This line left joins the input table with the table resulting from the query of the common Path. The first Step of the Path will serve as join attribute in the first iteration, but in subsequent iterations the common Path will serve as join attribute. The result of this line is a table with one more column than the input table.

Notice that line [6] makes use of the λ operator. However, the current λ specification does not conform to line [6] requirements. In particular, remark that the output of the λ operator is a one-column table (named after Path). However, besides this column, it is necessary another column that allows the left join with the input table. Accordingly, in order to inform the λ operator that another column is to maintain in the resulting table, a new parameter is necessary (*PathP*).

Furthermore, this parameter will also serve as the first Step of the query. This is necessary because the input table does not contain all attributes of the subject but only the attributes that result from the query process (e.g. the initial input table is projected into a one-column table). In that sense it is necessary to inform the λ operator about the initial join attribute. Therefore, the *PathP* parameter will be also used by the η operator as the first Step of the query process.

Therefore, while lines [2]-[5] require no changes, line [1] of the λ operator has to be changed to:

$$[1] \quad \lambda_{Path, PathP} T := \pi_{Path, PathP} \left(\rho_{Path / Path[\text{length}(Path)]} \left(\eta_{Path, PathP} T \right) \right)$$

This corresponds to the ultimate specification of the λ operator.

Example 6.14 – Querying KB through multiple tree-based represented Paths

Considering previous tree-based representation of Paths, $\varphi_{Paths_{O_1}}$ *SCI* would correspond to the annotated diagram of Figure 6.7:

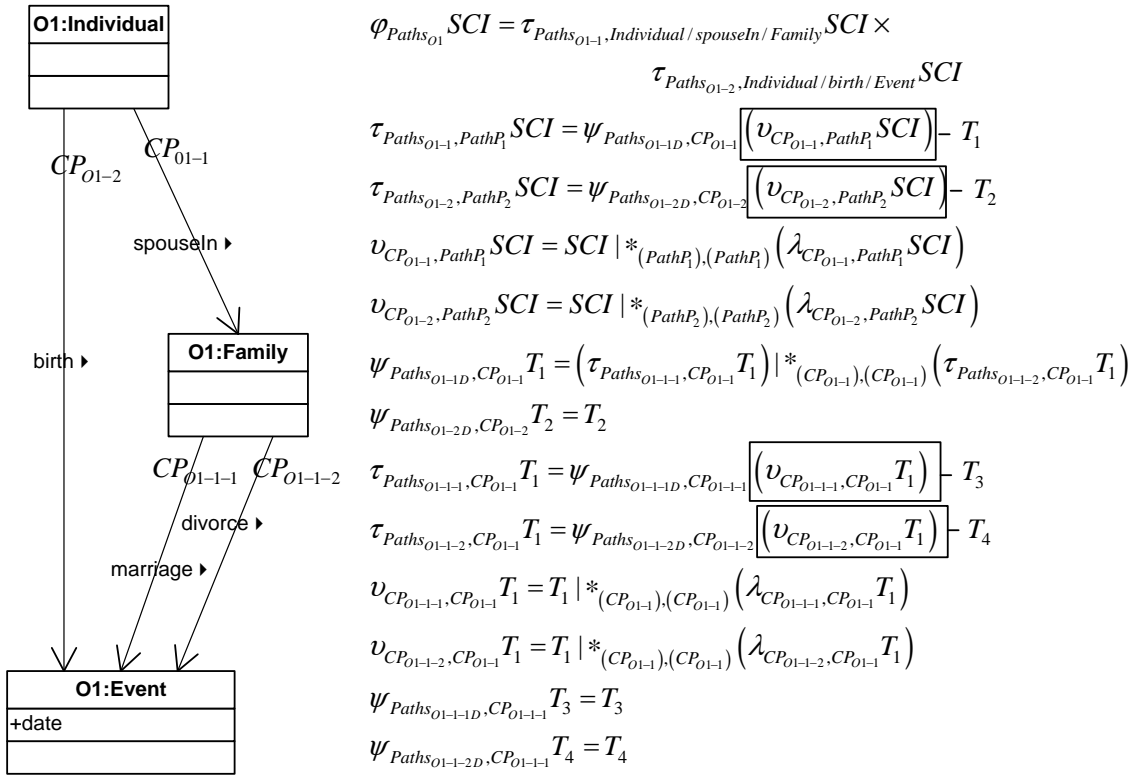


Figure 6.7 – Querying KB through Paths with and without common sub-Paths

The result of this phase is therefore an n-column table corresponding to the query of the tree-based representation of Paths. Table 6.10 presents the query result of previous Paths for instance $i_{1.1}$:

Table 6.10 - Query result of operation $\varphi_{Paths_{O_1}} i_{1.1}$

Individual/ spouseIn/Family	Individual/ spouseIn/Family/ marriage/Event/ date/Literal	Individual/ spouseIn/Family/ divorce/Event/ date/Literal	Individual/ birth/Event/ date/Literal
$f_{1.1}$	1796	1810	1769
$f_{1.2}$	1810		1769

For each branch in the tree-based representation of Paths a table column is delivered. However, in some circumstances not all delivered columns are required by the original Paths.

Example 6.15 – Useless query generated columns

Consider that, instead of previous four Paths, the first one is not specified in the SemanticBridge. Despite of this, the tree-based representation of the three Paths is the same in both cases. In fact, while the “Individual/spouseIn/Family” Path is not explicitly stated in the second case, it still appears in the tree-based representation of second case because it is the common sub-Path of both Path₂ and Path₃. As consequence, the

resulting table would include the “Individual/spouseIn/Family” attribute, even if it is not explicitly specified by original Paths.

Yet, because property values are accessed through the fully qualified Path, the existence of non-explicit Paths in the resulting query causes no problems and may be maintained in the resulting query.

6.2.2 Filter the knowledge base query

It is now time to filter the rows of the resulting query table according to the ConditionExpression defined in the SemanticBridge.

Because every source Path defined in the ConditionExpression has been also included in the tree-based representation and in the resulting query, the evaluation of ConditionExpressions runs for every row in the table.

Algebraically, this phase corresponds to a Select operation in the form of:

$$FQ := \sigma_{ConditionExpression} (\varphi_{Paths} SCI)$$

This table is referred as the Filtered Query (*FQ*) table.

Example 6.16 – Filtering queries

Consider that the following ConditionExpression is associated with the SemanticBridge that gave rise to previous table (Table 6.10):

$$\mathcal{K}_1 = \{Individual / spouseIn / Family / marriage / Event / date / Literal > 1800\}$$

The resulting table would be:

Table 6.11 - *FQ* table resulting from $\sigma_{\mathcal{K}_1} (\varphi_{Paths_{o1}} i_{1.1})$ operation

Individual/ spouseIn/Family	Individual/ spouseIn/Family/ marriage/Event/ date/Literal	Individual/ spouseIn/Family/ divorce/Event/ date/Literal	Individual/ birth/Event/ date/Literal
f _{1.2}	1810		1769

6.2.3 Create the target knowledge base instances

The transformation Service associated with the SemanticBridge will be executed for every row in the resulting table, creating a target ontology instance for each row.

As suggested and enumerated in 5.2.2, several transformation Services are envisaged. Some of the most commonly used are described next:

- CopyAttribute Service copies (no changes) source attribute instances into new target attribute instances. Its interface is depicted in Table 6.12.

Table 6.12 – CopyAttribute Service interface

Argument ID	Type	Semantics
Source Attribute	AttributePath	Source ontology attribute whose instances will be copied.
Target Attribute	AttributePath	Target ontology attribute in which instances will be created.

- CopyRelation Service transforms source property instances into target relations. Its interface is depicted in Table 6.13:

Table 6.13 – CopyRelation Service interface

Argument ID	Type	Semantics
Source Path	Path	Source ontology path for each path the bridge will be executed.
Target Path	RelationPath	Target ontology path to create.

The CopyRelation Service is a special Service concerning its role in the system. In particular, CopyRelation is responsible for the creation of relationships (relation instances) between target concept instances. It is understood as a built-in, seldom-evolving Service. Despite this categorization, its behaviour and application in PropertyBridges follows the same rules as any other Service. It will be deeply described, analyzed and deployed during the rest of this chapter. In that sense, do not consider this the ultimate description of the CopyRelation Service;

- Split Service transforms the instance of one source attribute into one instance of multiple target attributes. It takes the source attribute instance (i.e. Literal) and divides it by the Literals provided in the Separator parameter. Its interface is depicted in Table 6.14:

Table 6.14 – Split Service interface

Argument ID	Type	Semantics
Source Attribute	AttributePath	Source ontology attribute whose instances will be splitted.
Separators	ArrayOfLiterals	Literals or regular expressions to split by.
Target Attributes	ArrayOfAttributePaths	List of target attributes to create with splitted values.

- Concatenation Service concatenates several attribute instances into one target attribute instance. The strings defined through the Separators parameter are concatenated between every two source instances. Its interface is depicted in Table 6.15:

Table 6.15 – Concatenation Service interface

Argument ID	Type	Semantics
Source Attributes	ArrayOfAttributePath	List of source attributes whose instances will be concatenated.
Separators	ArrayOfLiterals	Literals to concatenate between attribute values.
Target Attribute	AttributePath	Target attributes to create with concatenated values.

- **RegularExpressionSubstring Service** creates a target attribute instance from the first occurrence in the text (of the source attribute instance) that matches the value in Regular Expression parameter. Its interface is depicted in Table 6.16:

Table 6.16 – RegularExpressionSubstring Service interface

Argument ID	Type	Semantics
Source Attribute	AttributePath	Source ontology attribute whose instances will be scanned.
Regular Expression	ArrayOfLiterals	Literals or regular expressions to split by.
Target Attributes	ArrayOfAttributePaths	List of target attributes to create with splited values.

- **CountProperties Services** counts the number of properties instances and creates the target attribute instance with the resulting value. Its interface is depicted in Table 6.17:

Table 6.17 – CountProperties Service interface

Argument ID	Type	Semantics
Source Path	Path	Source ontology path whose instances are counted.
Target Attribute	AttributePath	Target ontology attribute to create.

- **AttributeTableTranslation Service** transforms source attribute instances into target attribute instances according to the translation map provided in the external file specified by the File Location parameter. Its interface is depicted in Table 6.18:

Table 6.18 – AttributeTableTranslation Service interface

Argument ID	Type	Semantics
Source Attribute	AttributePath	Source ontology attribute whose instances will be transformed.
File location	FilePath	The location of the file containing the table.
Target Attribute	AttributePath	Target attributes to instantiate with translated value.

Some of these Services correspond to an empirically understandable transformation, which will be exemplified in the following annotated example. Others however, require further knowledge about their internal process and inter-relations. This is case of the Copy Instance and the Copy Relation Services, which capture and reflect a very important part of the work developed in the scope of this thesis. These two Services will be deeply described in section 6.3.

6.2.4 Example 6.17 – Execution process annotated example

The example presented in this section describes the execution process under the following perspectives:

- **ConceptBridges**, corresponding to the execution of CopyInstance Service;

- PropertyBridges that create target attribute instances, which corresponds to the execution of all Services except of CopyInstance and CopyRelation Services;
- PropertyBridges that create relationships between target concept instances, which corresponds to the execution of the CopyRelation Service.

Each of these SemanticBridges perspectives will be described and analyzed in the next three sections.

6.2.4.1 ConceptBridge

Consider the ontology mapping scenario depicted in Figure 6.8, where excerpts of TourinFrance [TourinFrance] (TIF namespace) and SIGRT [SIGRT] (SIGRT namespace) ontologies are being mapped.

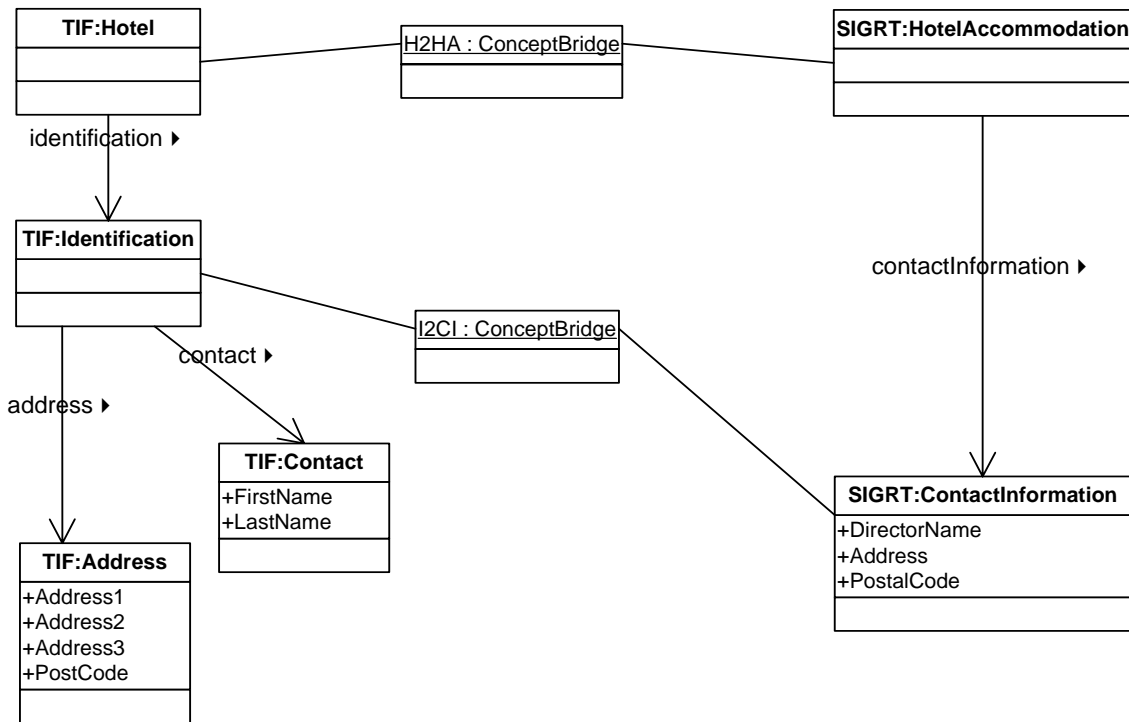


Figure 6.8 – ConceptBridges between excerpts of TourinFrance and SIGRT ontologies

The two minimally represented ConceptBridges of Figure 6.8, are fully specified in the following definition:

$$\mathcal{M}_{TS} = (TIF, SIGRT, \mathcal{T}_{TS}, \mathcal{B}_{TS}^C, \mathcal{B}_{TS}^P, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{B}_{TS}^C = \{H2HA, I2CI\}$$

$$H2HA = (TIF : Hotel, SIGRT : HotelAccommodation, \{ \}, \{ \}, \{ \})$$

$$I2CI = (TIF : Identification, SIGRT : ContactInformation, \{ \}, \{ \}, \{ \})$$

The execution engine performs ConceptBridges in an arbitrary order but always before any PropertyBridge. Consider the execution of previous ConceptBridges upon the following KB:

$$\mathcal{I}_{TIF} = \{h_1, h_2, i_1, i_2, i_3, c_1, c_2, c_3, c_4, a_1, a_2, a_3\}$$

$$inst_{\mathcal{C}_{TIF}} = \left\{ \begin{array}{l} Hotel(h_1), Hotel(h_2), \\ Identification(i_1), Identification(i_2), Identification(i_3), Identification(i_4), \\ Contact(c_1), Contact(c_2), Contact(c_3), Contact(c_4), \\ Address(a_1), Address(a_2), Address(a_3) \end{array} \right\}$$

Because no ConditionExpressions are specified for any of the neither ConceptBridges, nor sub bridge relations exists between them, their execution is straightforward. The resulting KB is:

$$\mathcal{I}_{SIGRT} = \{ha_1, ha_2, ci_1, ci_2, ci_3\}$$

$$inst_{\mathcal{C}_{SIGRT}} = \left\{ \begin{array}{l} HotelAccommodation(ha_1), HotelAccommodation(ha_2), \\ ContactInformation(ci_1), ContactInformation(ci_2), \\ ContactInformation(ci_3), ContactInformation(ci_4) \end{array} \right\}$$

The execution of ConceptBridge will give raise to a filled in transformation information table (TI^2) as represented in Table 6.19:

Table 6.19 – Table-based representation of TI^2 for previous ConceptBridges and KB

Source Concept Instance ID	Target Concept Instance ID	ConceptBridge
h ₁	ha ₁	H2HA
h ₂	ha ₂	H2HA
i ₁	ci ₁	I2CI
i ₂	ci ₂	I2CI
i ₃	ci ₃	I2CI

Once all ConceptBridges are executed no more concept instances will be created in the target knowledge base during this execution process.

6.2.4.2 PropertyBridge

Consider now that four PropertyBridges have been complementarily defined (Figure 6.9):

$$\mathcal{T}_{TS} = \{CopyRelation, CopyAttribute, Concatenation\}$$

$$\mathcal{B}_{TS}^P = \{names, addresses, pcode, i2ci\}$$

$$\diamond_{TS} = \{(I2CI, \{names, addresses, pcode\}), (H2HA, \{i2ci\})\}$$

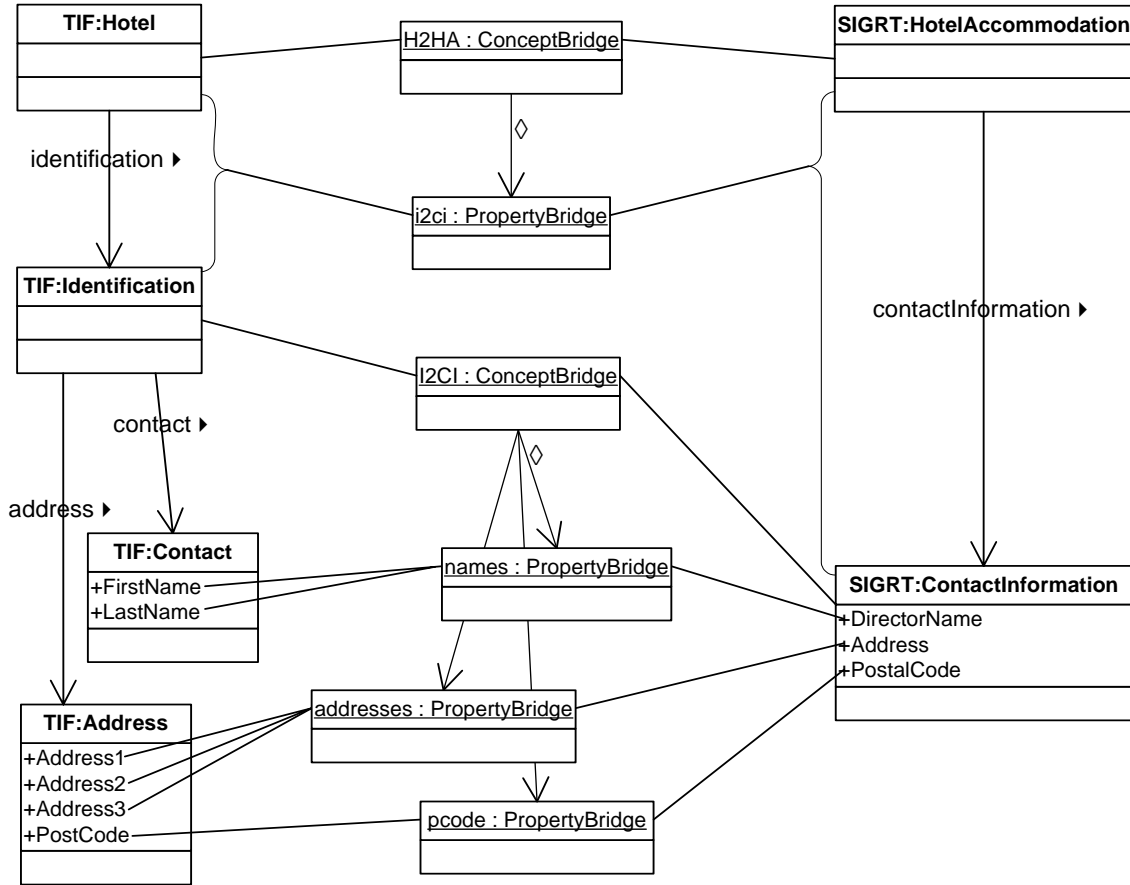


Figure 6.9 – Several SemanticBridges between TourinFrance and SIGRT ontologies

Additionally, consider the following properties instances (also represented in Figure 6.10):

$$\text{inst}_{\mathcal{P}_{TIF}} = \left\{ \begin{array}{l}
 \text{identification}(h_1, i_1), \text{identification}(h_1, i_2), \text{identification}(h_2, i_3), \\
 \text{contact}(i_1, c_1), \text{contact}(i_2, c_2), \text{contact}(i_3, c_3), \text{contact}(i_3, c_4), \\
 \text{address}(i_1, a_1), \text{address}(i_2, a_2), \text{address}(i_3, a_3), \\
 \text{FirstName}(c_1, \text{"John"}), \text{LastName}(c_1, \text{"Smith"}), \\
 \text{FirstName}(c_2, \text{"James"}), \text{LastName}(c_2, \text{"Ewing"}), \\
 \text{FirstName}(c_3, \text{"Otto"}), \text{LastName}(c_3, \text{"Halle"}), \\
 \text{FirstName}(c_4, \text{"Ralf"}), \text{LastName}(c_4, \text{"Frigs"}), \\
 \text{Address1}(a_1, \text{"245"}), \text{Address2}(a_1, \text{"22nd St."}), \text{Address3}(a_1, \text{"New York"}), \\
 \text{Address1}(a_2, \text{"43"}), \text{Address2}(a_2, \text{"Broad St."}), \text{Address3}(a_2, \text{"New York"}), \\
 \text{Address1}(a_3, \text{"24"}), \text{Address2}(a_3, \text{"Dorf Str."}), \text{Address3}(a_3, \text{"Berlin"}), \\
 \text{PostCode}(a_1, \text{"10166"}), \text{PostCode}(a_2, \text{"10004"}), \text{PostCode}(a_3, \text{"12529"})
 \end{array} \right\}$$

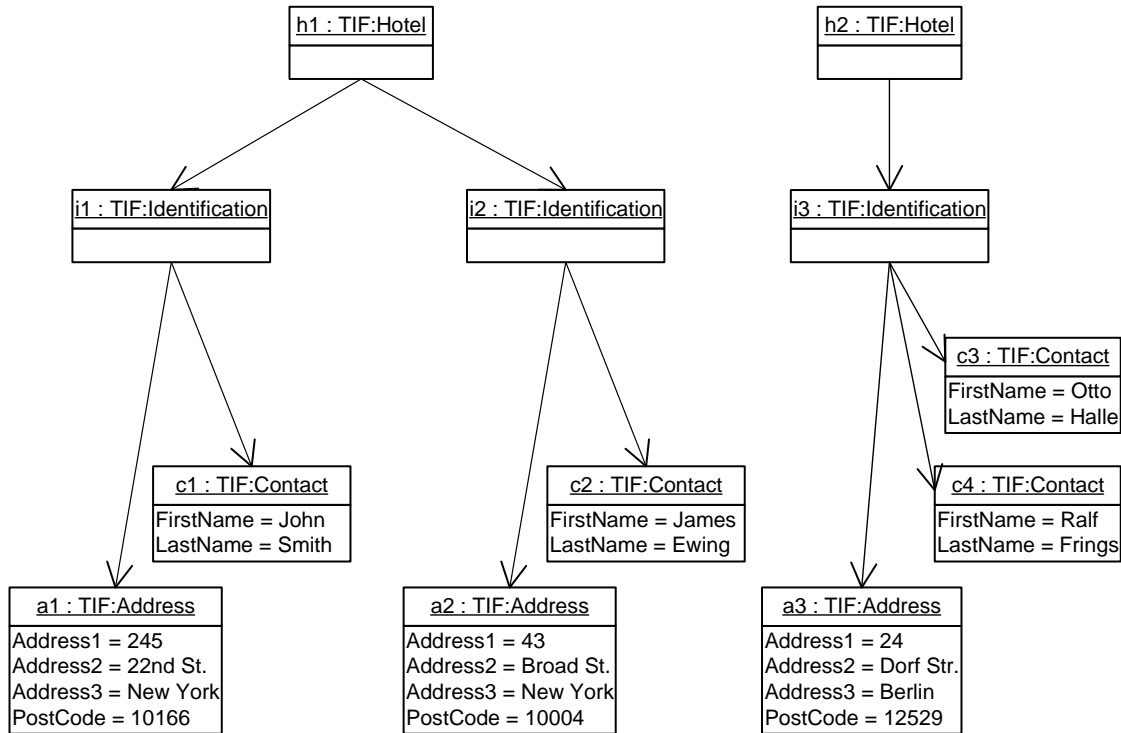


Figure 6.10 – Some instances of the TourinFrance ontology

The pcode PropertyBridge copies the TIF:Identification/contact/Contact/PostCode/Literal instances to PostalCode attribute instances of the current SIGRT:ContactInformation instance:

$$pcode = (\mathcal{W}_{TS.pcode}^s, \mathcal{W}_{TS.pcode}^t, \{ \}, CopyAttribute, \phi_{TS.pcode}^s, \phi_{TS.pcode}^t, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{W}_{TS.pcode}^s = \{ W_{TS.1}^s \}$$

$$W_{TS.1}^s = TIF : Identification / address / Address / PostCode / Literal$$

$$\mathcal{W}_{TS.pcode}^t = \{ W_{TS.1}^t \}$$

$$W_{TS.1}^t = SIGRT : ContactInformation / PostalCode / Literal$$

$$\phi_{TS.pcode}^s = \{ (W_{TS.1}^s, sourcePath) \}$$

$$\phi_{TS.pcode}^t = \{ (W_{TS.1}^t, targetPath) \}$$

Executing pcode in the scope of the ci₁ instance results in the following FQ table (Table 6.20):

Table 6.20 - FQ table resulting from the pcode execution in the scope of ci₁ instance

Identification/address/Address/PostCode/Literal
“10166”

Because the pcode PropertyBridge applies the CopyAttribute Service, it means that one target attribute instance will be created for each instance of source Path (i.e. Identification/address/Address/PostCode/Literal). The property instances resulting from the pcode PropertyBridge execution for all target concept instances are as follows:

$$inst\mathcal{P}_{SIGRT} = \{PostalCode(ci_1, "10166"), PostalCode(ci_2, "10004"), PostalCode(ci_3, "12529")\}$$

The addresses PropertyBridge aims to create TIF:ContactInformation/Address/Literal instances from the concatenation of Address1, Address2 and Address3 attributes instances, accessed from the instance of SIGRT:Identification. Furthermore, a space will be concatenated between Address1 and Address2 instances, and a comma and space between Address2 and Address3 instances. This description corresponds to the following specification:

$$addresses = (\mathcal{W}_{TS.addr}^s, \mathcal{W}_{TS.addr}^t, \{ \}, Concatenation, \phi_{TS.addr}^s, \phi_{TS.addr}^t, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{W}_{TS.addr}^s = \{W_{TS.2}^s, W_{TS.3}^s, W_{TS.4}^s\}$$

$$W_{TS.2}^s = TIF : Identification / address / Address / Address1 / Literal$$

$$W_{TS.3}^s = TIF : Identification / address / Address / Address2 / Literal$$

$$W_{TS.4}^s = TIF : Identification / address / Address / Address3 / Literal$$

$$\mathcal{W}_{TS.addr}^t = \{W_{TS.2}^t\}$$

$$W_{TS.2}^t = SIGRT : ContactInformation / Address / Literal$$

$$\phi_{TS.addr}^s = \{([\![W_{TS.2}^s, W_{TS.3}^s, W_{TS.4}^s]\!] , sourceAttributes)\}$$

$$\delta_{TIF.addr}^s = \{([\![", ", "]\!] , separators)\}$$

$$\phi_{TS.addr}^t = \{(W_{TS.2}^t, targetAttribute)\}$$

Executing addresses PropertyBridge in the scope of the ci_2 instance results in the following *FQ* table (Table 6.20):

Table 6.21 - *FQ* table resulting from the addresses execution in the scope of ci_2 instance

Identification/address/ Address/Address1/Literal	Identification/address/ Address/Address2/Literal	Identification/address/ Address/Address3/Literal
"43"	"Broad St."	"New York"

Accordingly, these three attribute instances will be concatenated into a unique string in the form of "43 Broad St., New York". Identical process occurs for each of the other instances of SIGRT:ContactInformation concept, resulting in the following instances:

$$inst\mathcal{P}_{SIGRT} = \left\{ \begin{array}{l} Address(ci_1, "245 22nd St., NewYork"), \\ Address(ci_2, "43 Broad St., New York"), Address(ci_3, "24 Dorf Str., Berlin") \end{array} \right\}$$

Identical process occurs for the names PropertyBridge. FirstName and LastName attributes should be concatenated into DirectorName attribute. FirstName and LastName properties should be accessed through the TIF:Identification/contact/Contact/FirstName/Literal and TIF:Identification/contact/Contact/LastName/Literal Paths respectively:

$$\begin{aligned}
 names &= (\mathcal{W}_{TS.names}^s, \mathcal{W}_{TS.names}^t, \{ \}, Concatenation, \phi_{TS.names}^s, \phi_{TS.names}^t, \{ \}, \{ \}, \{ \}, \{ \}) \\
 \mathcal{W}_{TS.names}^s &= \{W_{TS.5}^s, W_{TS.6}^s\} \\
 W_{TS.5}^s &= TIF : Identification / contact / Contact / FirstName / Literal \\
 W_{TS.6}^s &= TIF : Identification / contact / Contact / LastName / Literal \\
 \mathcal{W}_{TS.names}^t &= \{W_{TS.3}^t\} \\
 L_{TS.3}^t &= SIGRT : ContactInformation / DirectorName / Literal \\
 \phi_{TS.names}^s &= \{([\![W_{TS.5}^s, W_{TS.6}^s]\!], sourceAttributes)\} \\
 \delta_{TIF.names}^s &= \{([""], separators)\} \\
 \phi_{TS.names}^t &= \{(W_{TS.3}^t, targetAttribute)\}
 \end{aligned}$$

The execution of this PropertyBridge in the scope of ci_3 instance results in the following FQ table:

Table 6.22 - FQ table resulting from the names execution in the scope of ci_3 instance

Identification/contact/Contact/ FirstName/Literal	Identification/contact/Contact/ LastName/Literal
"Otto"	"Halle"
"Ralf"	"Frings"

For each row of previous table, a target instance will be created. The set of property instances resulting from this PropertyBridge execution for all target instances is therefore:

$$inst\mathcal{P}_{SIGRT} = \left\{ \begin{array}{l} DirectorName(ci_1, "John Smith"), DirectorName(ci_2, "James Ewing"), \\ DirectorName(ci_3, "Otto Halle"), DirectorName(ci_3, "Ralf Frings") \end{array} \right\}$$

In case of ci_3 instance, it corresponds to say that the hotel has two directors, which might be considered a semantic error. In case the domain expert decides that a unique director is admissible, it is necessary to apply cardinality constraints. This subject is addressed in section 6.4.

6.2.4.3 Inter-relation of instances

This section concerns the creation of relationships between target ontology concept instances. The CopyRelation Service is used for this purpose, but its functionality and internal details are very dependent on the CopyInstance Service:

- CopyInstance Service creates target ontology instance and the transformation information tuple in TI^2 ;
- CopyRelation Service inter-relates target concept instances according to the map provided by TI^2 and according to the source ontology property instances.

The PropertyBridge whose Service is the CopyRelation is \diamond -related with the ConceptBridge whose target concept is the domain of the relation to instantiate through the PropertyBridge.

In order to create relationships between target instances it is necessary to define the source ontology relation to copy from and the target ontology relation to instantiate. Such elements are specified through Paths. While the target ontology relation Path is mandatorilly a 1-Step Path, the source ontology Path may contain an arbitrary number of Steps.

Because the source relation Path has been included in the query, the resulting *FQ* table will have a column corresponding to the source Path instances. According to this *FQ* table, the Copy Relation transformation Service comprehends three stages:

1. Identify the pair of source concept instances whose relationship is to be copied;
2. Identify the pair of target instances resulting from previous source instances pair;
3. Create the relationship between target instances.

Considering the ontology mapping scenario of Figure 6.9 (below), the i2ci PropertyBridge aims to copy the relationships between TIF:Hotel and TIF:Identification instances, to relationships between SIGRT:HotelAccommodation and SIGRT:ContactInformation:

$$i2ci = (\mathcal{W}_{TS.i2ci}^s, \mathcal{W}_{TS.i2ci}^t, \{ \}, CopyRelation, \phi_{TS.i2ci}^s, \phi_{TS.i2ci}^t, \{ \}, \{ \}, \{ \}, \{ \}, \{ \})$$

$$\mathcal{W}_{TS.i2ci}^s = \{ W_{TS.7}^s \}$$

$$W_{TS.7}^s = TIF : Hotel / identification / Identification$$

$$\mathcal{W}_{TS}^t = \{ W_{TS.4}^t \}$$

$$W_{TS.4}^t = SIGRT : HotelAccommodation / contactInformation / ContactInformation$$

$$\phi_{TS.i2ci}^s = \{ (W_{TS.7}^s, sourcePath) \}$$

$$\phi_{TS.i2ci}^t = \{ (W_{TS.4}^t, targetPath) \}$$

The first stage operates upon the resulting *FQ* table. Because only one source Path is defined in the i2ci PropertyBridge, the resulting *FQ* table will be a 1-column table. Moreover, because the source Path is defined between TIF:Hotel and TIF:Identification, the resulting table will correspond to the relationships between the TIF:Hotel instance and one or more of TIF:Identification instances. In case of h_1 instance, the *FQ* table corresponds to Table 6.23:

Table 6.23 – *FQ* table resulting from the i2ci execution in the scope of ha_1 instance

Hotel/identification/Identification
i_1
i_2

According to the FQ table, the h_1 TIF:Hotel instance is related to i_1 and i_2 TIF:Identification instances. This information is the outcome of the first stage of the CopyRelation Service.

The second stage, known as correlation process, runs for every row in FQ table. In this stage it is necessary to determine the ID of the target concept instance that has been originated from the source concept instance defined in the table resulting from previous stage (e.g. i_1). For that, every table value is matched against the Source Concept Instance ID column of TI^2 , which in turn corresponds (maps) to the target concept instance ID that this stage is seeking for (Figure 6.11).

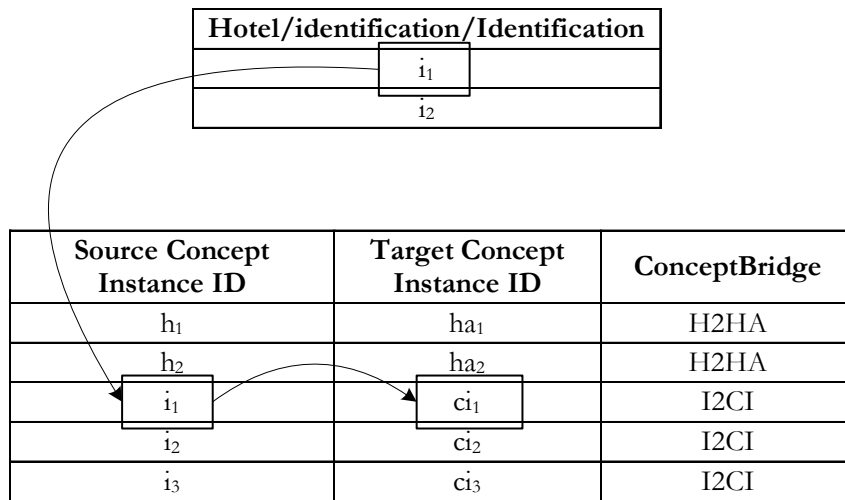


Figure 6.11 – Correlating source and target concept instances through FQ and TI^2

Third stage instantiates the target instance property value with the evaluated target concept instances. Accordingly, the i_2ci PropertyBridge execution, in the scope of all target concept instances, corresponds to the following target KB:

$$instP_{SIGRT} = \left\{ \begin{array}{l} contactInformation(ha_1, ci_1), contactInformation(ha_1, ci_2), \\ contactInformation(ha_2, ci_3) \end{array} \right\}$$

6.3 Extensional Specification

As defined so far, the CopyRelation process runs well for ontology mapping scenarios where one source concept instance originates a unique target concept instance (i.e. 1:1 semantic relations between concepts) or multiple concepts to one concept semantic relations (i.e. n:1 semantic relations between concepts) . However, if multiple target concept instances are created from the same source instance (i.e. 1:n semantic relations between concepts), the correlation between source and target instances becomes ambiguous. This ambiguity arises because two or more values in the Target Concept Instance column of TI^2 correspond to the same Source Concept Instance value.

6.3.1 Example 6.18 - ConceptBridges with 1:n cardinality

Consider the ontology mapping scenario presented in Figure 6.12, where SIGRT ontology is being mapped to TourinFrance ontology.

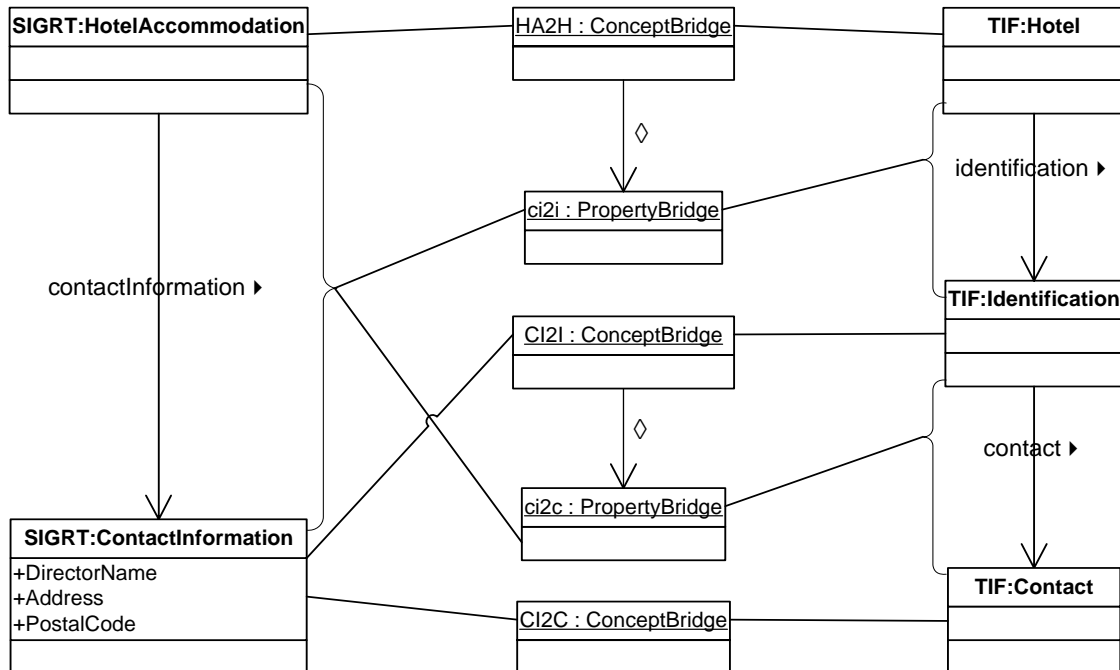


Figure 6.12 – Excerpts of SemanticBridges between SIGRT and TourinFrance ontologies

According to the inverse scenario (Figure 6.8) SIGRT:HotelAccommodation semantically relates to TIF:Hotel, and SIGRT:ContactInformation semantically relates to TIF:Identification, suggesting the specification of the HA2H and CI2I ConceptBridges.

Moreover, it is necessary to define a ConceptBridge responsible for the creation of instances of TIF:Contact. Because the DirectorName property is the entity that resembles more the semantics of TIF:Contact, the ConceptBridge should bridge SIGRT:ContactInformation to TIF:Contact.

Complementarily, it is necessary a PropertyBridge to create the relationships between TIF:Identification and TIF:Contact instances. Because TIF:Contact instances are created from SIGRT:ContactInformation instances, the source Path for the CopyRelation Service should be SIGRT:HotelAccommodation/contactInformation/ContactInformation.

Consider the following excerpt of the SIGRT knowledge base (corresponds to the set of concept instances created from TIF knowledge base in example presented in section 6.2.4):

$$\mathcal{I}_{SIGRT} = \{ha_1, ha_2, ci_1, ci_2, ci_3\}$$

$$inst\mathcal{C}_{SIGRT} = \left\{ \begin{array}{l} HotelAccommodation(ha_1), HotelAccommodation(ha_2), \\ ContactInformation(ci_1), ContactInformation(ci_2), \\ ContactInformation(ci_3) \end{array} \right\}$$

$$inst\mathcal{P}_{SIGRT} = \left\{ \begin{array}{l} contactInformation(ha_1, ci_1), contactInformation(ha_1, ci_2), \\ contactInformation(ha_2, ci_3), \\ DirectorName(ci_1, "John Smith"), DirectorName(ci_2, "James Ewing"), \\ DirectorName(ci_3, "Otto Halle"), DirectorName(ci_3, "Ralf Frings") \end{array} \right\}$$

The TI^2 table resulting from the execution of previous ConceptBridges corresponds to Table 6.24:

Table 6.24 – Table-based representation of TI^2

Source Concept Instance	Target Concept Instance	ConceptBridge
ha ₁	h _a	HA2H
ha ₂	h _b	HA2H
ci ₁	i _a	CI2I
ci ₂	i _b	CI2I
ci ₃	i _c	CI2I
ci ₁	c _a	CI2C
ci ₂	c _b	CI2C
ci ₃	c _c	CI2C

It is now time to execute PropertyBridges. The ci2i PropertyBridge is \diamond -related with HA2H, which means that it is executed in the scope of TIF:Hotel instances only (i.e. h_a and h_b, created from ha₁ and ha₂, respectively). Considering target instance h_a, the FQ table resulting from the ci2i PropertyBridge execution corresponds to Table 6.25:

Table 6.25 - FQ table for ci2i PropertyBridge, executed in the scope of h_a instance

HotelAccommodation/contactInformation/ContactInformation
ci ₁
ci ₂

Once the CopyRelation Service is associated with the ci2i PropertyBridge, the correlation process runs for every row of FQ table. The process aims to correlate the source instance ID value found in FQ table with the target instance created from the source instance. For first row in the table, the process finds two possible target instances (Figure 6.13).

According to the correlation process, instance ci₁ gave rise to both to i_a and c_a, which does not permits to decide univocally. While this is a quite simple scenario, it perfectly demonstrates the inaccuracy and lack of expressiveness of the process concerning 1:n semantic relations situations.

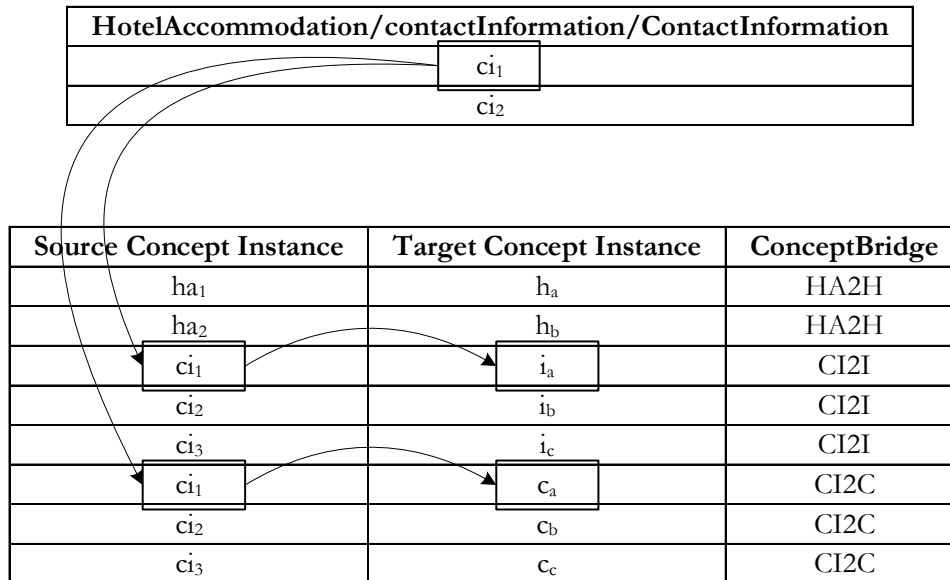


Figure 6.13 – Ambiguous correlation process

6.3.2 Analysis of the problem

The analysis of the problem results from observations carried out during user-based ontology mapping experiences. They showed that the 1:n concept to concept semantic relations are in fact the result of a combination of one concept to concept semantic relation plus many property to concept semantic relations. It has been noticed that multiple semantic relations between concepts are based on the existence or specific value of the instances of certain properties. In certain cases, even multiple properties instances are necessary to justify the creation of the target concept instance. Such cases are referred as n:1 properties to concept semantic relations. Other cases exist when one source property instance justify the creation of several target concept instances, known as 1:n property to concept semantic relations (5.2).

Notice that the same source property instance can give rise to multiple target instances, including target concept instances and target property instances.

Example 6.19 – Property to Concept semantic relations

Consider the ontology mapping scenario of Figure 6.14, where several semantic relations are depicted. In particular, notice that O1:Person/address/Literal is semantically related to (i) O2:Address; (ii) O2:Address/street/Literal, (iii) O2:Address/pobox/Literal, (iv) O2:Address/hasAddress/Address, (v) O2:WebAddress, (vi) O2:WebAddress/address/Literal, and (vii) O2:WebAddress/hasWebAddress/WebAddress.

While semantic relation between O1:Person and O2:Individual is directly supported in current SBO execution process (in this case the P2I ConceptBridge), the same of the other semantic relations are not yet supported. Hence the simple straight line to represent the semantic relations.

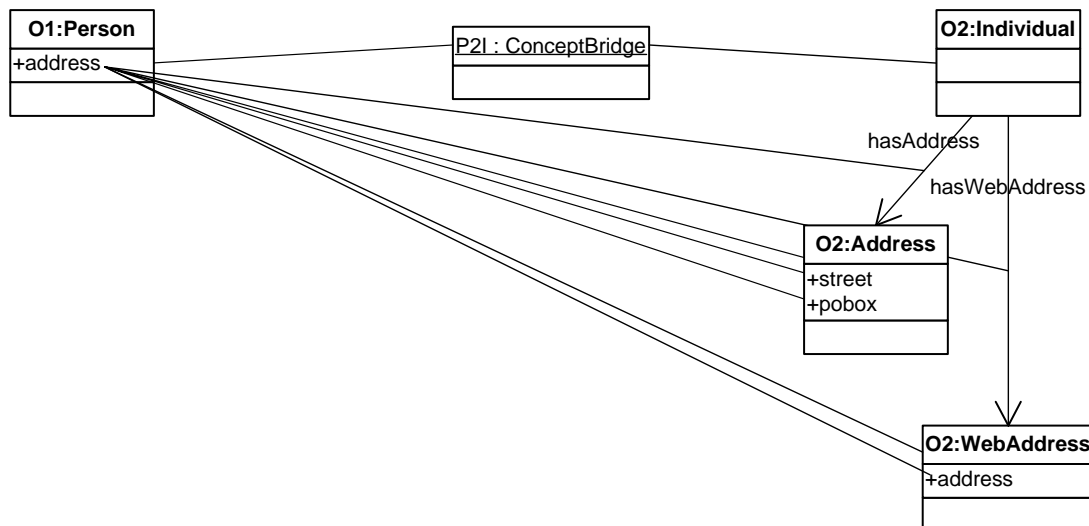


Figure 6.14 – 1:n Property to Concept semantic relations

As observed, it often occurs that target concepts are semantically related to source properties. In fact, in many situations, the simple existence of the property instance denotes the creation of the target concept. Other situations require that the property instance matches some value, or a certain number of instances exist.

Example 6.20 – Constrained Property to Concept semantic relations

Considering the scenario presented in Example 6.19, it empirically makes sense to create O2:Address concept instances in case O1:Person/address/Literal instance exists. Instead, the creation of O2:WebAddress instances might be constrained by the value of O2:Person/address/Literal instances. For example, it might be stated that an instance of O2:WebAddress is created if O2:Person/address/Literal instance values matches a URI pattern.

Two approaches naturally arise as potential solutions:

1. The creation of a new subclass of SemanticBridge, respecting the property-to-concept semantic relation. This approach is based on the entity type dimension of semantic relations, and promotes its importance. Concerning the adoption of this approach the following remarks arise:
 - 1.1 Clear ontological distinction according to the entity type dimension;
 - 1.2 New transformation Service required;
 - 1.3 More complexity in inter-relating SemanticBridges;
2. The adoption of an enhanced transformation Service permitting the specification of complementary arguments. This approach is based on the transformation dimension of semantic relations, and promotes its relevance. Concerning the adoption of this approach the following remarks arise:
 - 2.1 Clear ontological classification according to the transformation dimension;
 - 2.2 Evolution of the transformation Service is required;
 - 2.3 No changes of inter-relations between SemanticBridges are necessary.

Because both approaches are mutually exclusive, a decision has to be drawn based on the advantages and disadvantages mentioned in prior remarks. For that, a brief analysis has been done:

- Ontological clarification:
 - The first approach distinguishes more types of semantic relations, which does not directly represent benefits. In fact, under the application point-of-view, fine grained ontologies are not always better than more coarse grained ontologies. Fine grained ontologies often lead to a large and unpractical number of concepts.

Example 6.21 – Fine vs. Coarse grained ontology

Considering the cardinality dimension, a fine grained version of SBO has been presented in [Maedche *et al.*, 2002b] aiming to describe the ontology mapping domain of knowledge. However, this dimension has been completely abstracted under the transformation dimension in [Maedche *et al.*, 2002a] in order to meet the ontology mapping system application requirements (refer to 5.4.3);

- The second approach suggests the clarification of SemanticBridges according to the transformation dimension, which has been previously adopted in respect to the abstraction of the cardinality dimension (5.4.3). Promoting the transformation dimension in disfavor of the entity type dimension would promote consistency of the decisions in modeling SBO;
- Transformation service:
 - The first approach requires developing a new service in respect to the type of entities semantically related through the bridge. Moreover, adopting this approach, the competency of creation of target concept instance is duplicated in two types of SemanticBridges;
 - The second approach does not require a new transformation service, but requires the inclusion of functionalities permitting to define the source properties;
- Inter-relations between SemanticBridges concerns the changes in type and semantic of relations between the classes type of SemanticBridges. The following remarks arise:
 - The first approach suggests the specification of more types of semantic relations, which would cause an increased number of inter-relation types. Moreover, because the (hypothetic) property-to-concept SemanticBridge relates properties with concepts, its semantics may cause ambiguity in the stable inter-relation rules established between SBO SemanticBridges. Some ontology mapping approaches suggest the application of *skolem terms* [Dou *et al.*, 2002; Russel & Norvig, 1995], which requires nested relationships between ConceptBridges (refer to 5.3), affecting expressivity, declarativity, and clarity of the mapping document. While the *skolem term* based approach makes sense on systems based on inference engines, it tends to be hard to manage at semantic bridging and hard to track at execution phase;
 - The second approach suggests the maintenance of the SemanticBridges types, preventing further inter-relation types.

Table 6.26 summarizes the analysis described in previous paragraphs.

Table 6.26 – Summary of the Property-to-concept semantic relations approaches

Analyzed parameters	First Approach	Second Approach
1. Ontological clarification		
1.1 Dimension oriented	Entity type	Transformation
1.2 Ontological decision	Inconsistent	Consistent
2. Transformation Service requirements	New Service	New functionalities
3. SemanticBridges inter-relations		
3.1 Inter-relations required	New relations	No new relations
3.2 Ontological decision	Ambiguous	Clear

It is now possible to choose one of the approaches according to a set of indicators resulting from the systematized analysis. Indeed, from previous descriptions and previous table, second approach stands as the more advantageous and less disadvantageous, leading to its adoption.

As consequence, the ConceptBridge concept and SBO fundamental characteristics are maintained, including the inter-relations between ConceptBridge and between ConceptBridges and PropertyBridges. In fact, the CopyInstance Service is the only component requiring modifications.

6.3.3 Developed approach

Notice that the problem is not about how to create multiple target concept instances from the same source concept instance, but how to determine the correct target concept instance according to the source concept instance. I.e. how to refer to a concept instance that does not really exists as concept instance.

This issue assumes special importance for CopyRelation Service, which needs to evaluate source-target correlations based on TI^2 table, filled in by ConceptBridge executions (i.e. by the CopyInstance Service). The CopyRelation Service dependency from the CopyInstance Service is therefore evident and of fundamental importance for the execution process, motivating the coordination between both Services.

The developed approach is based on the notion of perspective of the source concept instance. Each source concept instance has multiple perspectives depending on the values of the properties selected to uniquely identify the concept instance perspective. This is referred as Extensional Specification⁴³.

⁴³ While the ontology is the intentional part of the knowledge base, the ontology entities instances correspond to the extensional part.

Example 6.22 – Different perspective of the same concept instance

Once again, considering the scenario presented in Example 6.19, one might say that certain instance of O1:Person will give rise to an instance of O2:Address under the perspective that an instance of O1:Individual/address/Literal property exists. The same source concept instance gives rise to an instance of O2:WebAddress under the perspective that the text of the O1:Individual/address/Literal property instance matches an URI pattern.

Once the properties values match the perspective requirements, the target concept instance is created and the properties values associated with the transformation information. The pair constituted by the source instance ID and the Extensional Specification forms a primary (unique) key in the TI^2 permitting the identification of the concept instance perspective that gives raise to every target instance. Moreover, because such information is present in the knowledge base it is possible to reconstruct it during the execution process whenever is needed, providing the means to refer to that concept instance perspective and therefore access the ID of the unique target concept instance created from that perspective.

Accordingly, it is necessary to incorporate the extensional specification features in the CopyInstance Service and coordinate the process with other Services, especially with the CopyRelation Service.

The extensional specification step is potentially based on arbitrary combination of multiple Path constraints. ConditionExpression features these competencies in perfection, arising as a strong candidate mechanism to represent the extensional specification. In particular, notice that:

- No new SBO entity would be required. ConditionExpression entity provides the mechanisms to represent complex arbitrary combination of properties instances and constant expressions;
- No changes are required in the query knowledge base process, since ConditionExpressions are based on Paths which are already addressed in the query process (refer to 6.2.1);
- No changes are required in filtering the result of the knowledge base query. Because the filtering process is based on ConditionExpressions, the ConditionExpression representing the extensional specification is just another one;
- Simplified evolution of SBO and execution process, since the constraints and extensional specification grounds on the same representation entity;
- Small changes in the CopyInstance and CopyRelation Service.

CopyInstance and CopyRelation Service need to be modified to include the ConditionExpressions of the Extensional Specification.

6.3.3.1 ConceptBridge and PropertyBridge

Because the CopyInstance Service has been made implicit in ConceptBridges, new Service arguments need to be explicitly included in the ConceptBridge specification.

Unlike in ConceptBridges, Services in PropertyBridges are explicit. The association between Service arguments and respective values is made through the ϕ^s and ϕ^t functions, which depend on the Service specification. At first sight, the Extensional Specification argument might be understood just like any other argument of the Service whose type is ConditionExpression. However, both ϕ^s and ϕ^t functions relate Paths to arguments only, which excludes ConditionExpressions. In that sense, the PropertyBridge specification requires further changes.

Despite distinct structures, ConceptBridge and PropertyBridge have some elements in common. One of these is the set of ConditionExpressions element, denoted by \mathcal{K} :

$$B^C := (c^s, c^t, Q, \delta^s, \delta^t, \mathcal{K})$$

$$B^P := (\mathcal{L}^s, \mathcal{L}^t, Q, S, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{K})$$

Notice that the ConditionExpression of the Extensional Specification argument is a full-fledged ConditionExpression, i.e. it constrains the execution process as any other ConditionExpression. In that sense, the \mathcal{K} argument might be used to convey ConditionExpressions to the SemanticBridge, without any side-effect. In order to assign specific ConditionExpressions to the Extensional Specification arguments, a specific function would be applied, as for the Paths and Literals arguments through the ϕ^s , ϕ^t , δ^s and δ^t functions respectively. This approach would correspond to the following specifications:

$$B^C := (c^s, c^t, Q, \delta^s, \delta^t, \mathcal{K}, \chi)$$

$$B^P := (\mathcal{L}^s, \mathcal{L}^t, Q, S, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{K}, \chi)$$

where:

- \mathcal{K} is the set of both general and Extensional Specification ConditionExpressions;
- χ is the function that assigns ConditionExpressions to the Extensional Specification arguments.

This approach has the following advantages:

- The same approach is adopted for both types of SemanticBridges;
- The function-based argument assignment approach is extended;
- The Service specification is modestly changed;

- The execution process is not changed;

and disadvantage:

- Potential ambiguity concerning the classification of ConditionExpressions, either referring to the Extensional Specification or as general ConditionExpressions.

In order to cope with the ambiguity problem of previous approach, another approach suggests the specification of a new SemanticBridge argument concerning the separate representation of the ConditionExpressions representing the Extensional Specification. Because multiple arguments might make use of these ConditionExpressions, a function would be necessary to assign every ConditionExpression to the proper Service argument, as suggested in prior approach. This approach would then be reflected in the following specifications:

$$B^C := (c^s, c^t, Q, \delta^s, \delta^t, \mathcal{K}, \mathcal{X}, \chi)$$

$$B^P := (\mathcal{L}^s, \mathcal{L}^t, Q, S, \phi^s, \phi^t, \delta^s, \delta^t, \mathcal{K}, \mathcal{X}, \chi)$$

where:

- \mathcal{X} is the set of ConditionExpressions concerning with the Extensional Specification;
- χ is the function that assigns ConditionExpressions to the Extensional Specification arguments.

The following advantages are envisaged:

- The same approach is adopted for both types of SemanticBridges;
- The function-based argument assignment approach is extended;
- The Service specification is modestly changed;
- Clear and univocal expressivity;

along with the following disadvantage:

- The filtering phase of the execution process will take not only the ConditionExpressions in \mathcal{K} but also the ConditionExpressions in \mathcal{X} (i.e. $\mathcal{K} \cup \mathcal{X}$).

The last described solution has been adopted in disfavor of first approach since it has been considered that the clarity and univocal expression of semantic relations is more important than the change in the execution process.

6.3.3.2 Semantics of the Extensional Specification arguments

Despite their equal name, semantics of the Extensional Specification argument of CopyInstance and CopyRelation Services differ substantially. In CopyInstance Service, the argument aims to determine and evaluate the constraints of properties instances necessary to define a new and unique

perspective of the concept instance. Such a new target instance is created according to such unique perspective (Table 6.27).

Table 6.27 – CopyInstance Service parameters

Argument ID	Type	Semantics
Source Concept	Concept	Source ontology concept to copy.
Extensional Specification	ConditionExpression	Expression representing the properties and constraints applied in characterizing the source concept instance.
Target Concept	Concept	Target ontology concept to create.

In CopyRelation Service instead, it refers to a specific source concept instance perspective in order to correlate it with the target concept instance created from it (Table 6.28).

Table 6.28 – CopyRelation Service parameters

Argument ID	Type	Semantics
Source Path	Path	Source ontology path for each path the bridge will be executed.
Extensional Specification	ConditionExpression	Expression representing the properties and constraints applied in characterizing the source concept instance.
Target Path	RelationPath	Target ontology path to create.

Accordingly, the TI^2 table is the inter-relation mechanism between both Services:

- The CopyInstance Service writes the TI^2 . For every target instance created it will attach the extensional specification information to the transformation information tuple;
- The CopyRelation Service reads the TI^2 . For every relation to create, it complements the source concept instance ID with the extensional specification information in order to define a unique characterization of a source concept instance.

Notice that the read operation is not exclusive of the CopyRelation Services but instead it may be used by any Service that requires the Extensional Specification feature.

As consequence, the transformation information tuple assumes the following form:

$$TI := \left(\begin{array}{l} source_concept_instance, extensional_specification_expression, \\ target_concept_instance, concept_bridge \end{array} \right)$$

where:

- $source_concept_instance$, $target_concept_instance$ and $concept_bridge$ corresponds to the elements as defined in section 6.1;
- $extensional_specification_expression$ is a structure in the form of:

$$\begin{aligned} \langle instantiated_condition_expression \rangle &::= \langle and \rangle | \langle or \rangle | \langle xor \rangle | \langle not \rangle | \langle instantiated_comparison \rangle \\ \langle and \rangle &::= and(\langle instantiated_condition_expression \rangle \{ ", " \} \langle instantiated_condition_expression \rangle) \end{aligned}$$

$$\begin{aligned} \langle \text{or} \rangle &::= \text{or} \left(\langle \text{instantiated_condition_expression} \rangle \{ ", " \langle \text{instantiated_condition_expression} \rangle \} \right) \\ \langle \text{xor} \rangle &::= \text{xor} \left(\langle \text{instantiated_condition_expression} \rangle, \{ ", " \langle \text{instantiated_condition_expression} \rangle \} \right) \\ \langle \text{not} \rangle &::= \text{not} \left(\langle \text{instantiated_condition_expression} \rangle \right) \\ \langle \text{instantiated_comparison} \rangle &::= \langle \text{instance_operand_1} \rangle \langle \text{OPERATOR} \rangle \langle \text{instance_operand_2} \rangle \\ \langle \text{instance_operand_1} \rangle &::= \langle \text{path_value} \rangle \\ \langle \text{instance_operand_2} \rangle &::= \langle \text{path_value} \rangle | \langle \text{LITERAL} \rangle \\ \langle \text{path_value} \rangle &::= " \langle \text{PATH} \rangle, \langle \text{VALUE} \rangle " \end{aligned}$$

where:

- $\langle \text{PATH} \rangle$, $\langle \text{OPERATOR} \rangle$ and $\langle \text{LITERAL} \rangle$ are as described in 5.4.5;
- $\langle \text{VALUE} \rangle$ is the value for column $\langle \text{PATH} \rangle$ in the *FQ* table

The extension specification expression element of the transformation information tuple is therefore an instantiated *ConditionExpression* such its tokens $\langle \text{operand_1} \rangle$ and $\langle \text{operand_2} \rangle$ are transformed into $\langle \text{instance_operand_1} \rangle$ and $\langle \text{instance_operand_2} \rangle$ respectively, representing not only the Paths operand but also the instances of the Path as found in *FQ* table.

Because the *extensional_specification_information* relays on *ConditionExpression* representation, the extensional specification mechanism is structure-oriented. As referred for the *ConditionExpression* (5.4.5), this is not a limitation but a feature. In fact, the same set of comparison elements serve to create distinct source instances perspectives due to the fact that the comparison between *instantiated_condition_expressions* is performed by the structure and not by the logical value of comparisons. This feature provides the mechanism to create multiple target instances from the same logical source instance perspective.

Example 6.23 – Structural interpretation of ConditionExpressions

Because the two following *ConditionExpressions* are considered different, they define distinct perspectives of the same source instance:

$$\begin{aligned} K_3 &= \left\{ \text{and} \left(\begin{array}{l} \text{TIF : Hotel / identification / Identification} == i_{\text{TIF.1}}, \\ \text{TIF : Hotel / identification / Identification} == i_{\text{TIF.1}} \end{array} \right) \right\} \\ K_4 &= \left\{ \text{and} \left(\text{TIF : Hotel / identification / Identification} == i_{\text{TIF.1}} \right) \right\} \end{aligned}$$

These two logically equivalent *ConditionExpressions* give rise to two distinct *instantiated_condition_expressions* that do not match between them, even if they always evaluate to the same logical value.

6.3.3.3 Extensional Specification and Description logics

The extensional specification step provides the mechanism to distinguish and address between different perspectives of the same (source) instance. Such approach can be understood as the virtual creation of multiple source instances. Extensional specification has closely resemblances with Description Logics (DL) as applied in Semantic Web ontology representation languages, such as OIL, DAML, DAML+OIL and OWL. Basically, DL-based ontology representation languages exploit characteristics of concepts to refine its classification. Generically, sub-classes are defined by constraining the values and characteristics of the super-class. This serves to explicitly and semantically describe the domain of knowledge and to infer the better classification (type) of an instance according to its property values.

The extensional specification mechanism does not intent to create new source concepts but instead to extensionally define virtual instances and access to its specification.

6.3.4 Example 6.24 - Extensional specification annotated example

Consider the mapping scenario of Figure 6.15, where SIGRT ontology is being mapped to TourinFrance ontology.

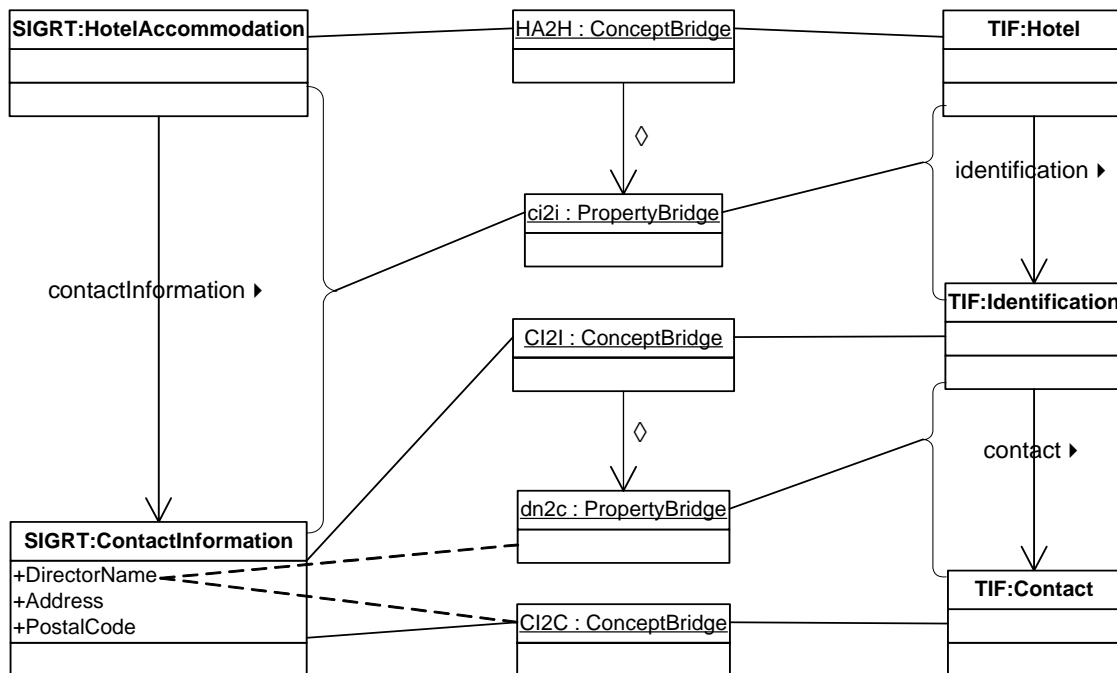


Figure 6.15 – Excerpt of the SIGRT-TIF mapping scenario

This UML diagram is underspecified due to the complexity to represent all Services arguments, including the Extensional Specification argument. The diagram is provided to complement the understanding of the following $\mathcal{M}_{SIGRT-TIF}$ specification, which indeed represents all details of the envisaged semantic bridging. PropertyBridges are also presented during this section.

$$\begin{aligned}
 \mathcal{M}_{SIGRT-TIF} &= \left(SIGRT, TIF, \mathcal{T}_{ST}, \mathcal{B}_{ST}^C, \mathcal{B}_{ST}^P, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \diamond_{ST}, \{ \} \right) \\
 \mathcal{B}_{ST}^C &= \{ HA2H, CI2I, CI2C \} \\
 HA2H &= \left(SIGRT : HotelAccommodation, TIF : Hotel, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \} \right) \\
 CI2I &= \left(SIGRT : ContactInformation, TIF : Identification, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \} \right) \\
 CI2C &= \left(SIGRT : ContactInformation, TIF : Contact, \{ \}, \{ \}, \{ \}, \{ \}, \mathcal{X}_{CI2C}, \mathcal{X}_{CI2C} \right) \\
 \mathcal{X}_{CI2C} &= \{ K_{CI2C} \} \\
 K_{CI2C} &= W_{ST.1}^s \equiv W_{ST.1}^s \\
 \mathcal{X}_{CI2C} &= \{ (K_{CI2C}, extensionalSpecification) \} \\
 W_{ST.1}^s &= SIGRT : ContactInformation / DirectorName / Literal
 \end{aligned}$$

Three ConceptBridges are defined between SIGRT and TourinFrance ontologies. While HA2H and CI2I ConceptBridges do not present any new aspects, the CI2I ConceptBridge makes use of the Extensional Specification mechanism, represented in the diagram by the dashed line. The Extensional Specification mechanism is further used in dn2c PropertyBridge, to create instances of TIF:Identification/contact/Contact relation from the SIGRT:ContactInformation/DirectorName/Literal source attribute instances.

According to the domain expert, one TIF:Contact instance is created for each instance of SIGRT:ContactInformation/DirectorName/Literal attribute. In that sense, each new TIF:Contact arises from each SIGRT:ContactInformation instance perspective such its SIGRT:ContactInformation/DirectorName/Literal instance is unique. The property instance is attached to the transformation information tuple, wrapped by the ConditionExpression structure.

Consider once again the following SIGRT knowledge base:

$$\begin{aligned}
 \mathcal{I}_{SIGRT} &= \{ ha_1, ha_2, ci_1, ci_2, ci_3 \} \\
 instC_{SIGRT} &= \left\{ \begin{array}{l} HotelAccommodation(ha_1), HotelAccommodation(ha_2), \\ ContactInformation(ci_1), ContactInformation(ci_2), \\ ContactInformation(ci_3) \end{array} \right\} \\
 instP_{SIGRT} &= \left\{ \begin{array}{l} contactInformation(ha_1, ci_1), contactInformation(ha_1, ci_2), \\ contactInformation(ha_2, ci_3), \\ DirectorName(ci_1, "John Smith"), DirectorName(ci_2, "James Ewing"), \\ DirectorName(ci_3, "Ralf Frings"), DirectorName(ci_3, "Otto Halle") \end{array} \right\}
 \end{aligned}$$

Because HA2H has no \mathcal{K} nor \mathcal{X} ConditionExpressions, for every source concept instance of SIGRT:HotelAccommodation one TIF:Hotel instance is created. Accordingly, no Extensional Specification information is attached to the respective transformation information tuples. Analogous process occurs for CI2I ConceptBridge. For the CI2C ConceptBridge instead, \mathcal{X}

ConditionExpressions are specified, which further require generation of the corresponding FQ table. The execution of CI2C ConceptBridge in the scope of ci_3 instance will call:

$$FQ = \sigma_{K_{CI2C}} \left(\varphi_{\{ContactInformation / DirectorName / Literal, \{\}\}} ci_3 \right)$$

which generates the following table:

Table 6.29 - FQ table for CI2C ConceptBridge and ci_3 instance

ContactInformation/DirectorName/Literal
“Ralf Frings”
“Otto Halle”

For every row of previous table, a source concept instance perspective will be created in TI^2 , providing the information to create a new target TIF:Contact instance. It corresponds to the last two lines of TI^2 (Table 6.29).

Table 6.30 - TI^2 resulting from execution of previous ConceptBridges upon the KB

Source Concept Instance ID	Extensional Specification	Target Concept Instance ID	ConceptBridge
ha ₁		h _a	HA2H
ha ₂		h _b	HA2H
ci ₁		i _a	CI2I
ci ₂		i _b	CI2I
ci ₃		i _c	CI2I
ci ₁	SIGRT:ContactInformation/DirectorName/Literal==”John Smith”	c _a	CI2C
ci ₂	SIGRT:ContactInformation/DirectorName/Literal==”James Ewing”	c _b	CI2C
ci ₃	SIGRT:ContactInformation/DirectorName/Literal==”Ralf Frings”	c _c	CI2C
ci ₃	SIGRT:ContactInformation/DirectorName/Literal==”Otto Halle”	c _d	CI2C

Consider now the rest of $\mathcal{M}_{SIGRT-TIF}$ mapping specification, in which the two generically represented PropertyBridges of Figure 6.15 are fully defined:

$$\begin{aligned} \mathcal{T}_{ST} &= \{CopyRelation\} \\ \mathcal{B}_{ST}^P &= \{ci2i, ci2c\} \\ \diamond_{ST} &= \{(HA2H, \{ci2i\}), (CI2I, \{ci2c\})\} \end{aligned}$$

$$ci2i = \left(\mathcal{W}_{ST-ci2i}^s, \mathcal{W}_{ST-ci2i}^t, \{ \}, CopyRelation, \phi_{ST-ci2i}^s, \phi_{ST-ci2i}^t, \{ \}, \{ \}, \{ \}, \{ \}, \{ \}, \{ \} \right)$$

$$\mathcal{W}_{ST-ci2i}^s = \{ W_{ST.2}^s \}$$

$$\mathcal{W}_{ST-ci2i}^t = \{ W_{ST.1}^t \}$$

$$\phi_{ST-ci2i}^s = \{ (W_{ST.2}^s, sourcePath) \}$$

$$\phi_{ST-ci2i}^t = \{ (W_{ST.1}^t, targetPath) \}$$

$$W_{ST.2}^s = SIGRT : HotelAccommodation / contactInformation / ContactInformation$$

$$W_{ST.1}^t = TIF : Hotel / identification / Identification$$

$$ci2c = \left(\mathcal{W}_{ST-ci2c}^s, \mathcal{W}_{ST-ci2c}^t, \{ \}, CopyRelation, \phi_{ST-ci2c}^s, \phi_{ST-ci2c}^t, \{ \}, \{ \}, \{ \}, \mathcal{X}_{CI2C}, \mathcal{X}_{ci2c} \right)$$

$$\mathcal{W}_{ST-ci2c}^s = \{ W_{ST.1}^s \}$$

$$\mathcal{W}_{ST-ci2c}^t = \{ W_{ST.2}^t \}$$

$$\phi_{ST-ci2c}^s = \{ (W_{ST.1}^s, sourcePath) \}$$

$$\phi_{ST-ci2c}^t = \{ (W_{ST.2}^t, targetPath) \}$$

$$\mathcal{X}_{ci2c} = \{ (K_{CI2C}, extensionalSpecification) \}$$

$$W_{ST.1}^s = SIGRT : ContactInformation / DirectorName / Literal$$

$$W_{ST.2}^t = TIF : Identification / contact / Contact$$

The novelty of this example is concentrated on the ci2c PropertyBridge. In this PropertyBridge the SIGRT:ContactInformation/contactInformation/Literal source attribute is to be copied to TIF:Identification/contact/Contact relation. This source ontology Path is used to uniquely identify a specific source instance perspective that created the target concept instance that is the object of the relationship to create. The referred Path is therefore used in both the *sourcePath* argument and in the ConditionExpression defined for the *extensionalSpecification* argument. For that, the *FQ* table should be evaluated once again according to:

$$FQ = \sigma_{K_{CI2C}} \left(\varphi_{(ContactInformation / DirectorName / Literal, \{ \})} ci_3 \right)$$

which evaluates exactly to the same *FQ* table evaluated for the CI2C ConceptBridge (Table 6.31). This is due to the fact that the same Extensional Specification ConditionExpression is used and applied to the same source concept instance.

Table 6.31 - *FQ* table for ci2c PropertyBridge upon ci_3 instance

ContactInformation/DirectorName/Literal
“Ralf Frings”
“Otto Halle”

For every row of the *FQ* table an instantiated ConditionExpression is generated, corresponding to Table 6.32:

Table 6.32 – Instantiated ConditionExpression for Extensional Specification

Extensional Specification
SIGRT:ContactInformation/DirectorName/Literal=="Ralf Frings"
SIGRT:ContactInformation/DirectorName/Literal=="Otto Halle"

To this table, a column is added to incorporate the source concept instance (Table 6.33):

Table 6.33 – Complete characterization of source instance perspective

Source Instance ID	Extensional Specification
ci ₃	SIGRT:ContactInformation/DirectorName/Literal=="Ralf Frings"
ci ₃	SIGRT:ContactInformation/DirectorName/Literal=="Otto Halle"

It is now possible to uniquely identify the pair formed by the source instance perspective and the target concept instance that has been created from it. Figure 6.16 depicts this process for the source instance perspective described by the first line of previous table.

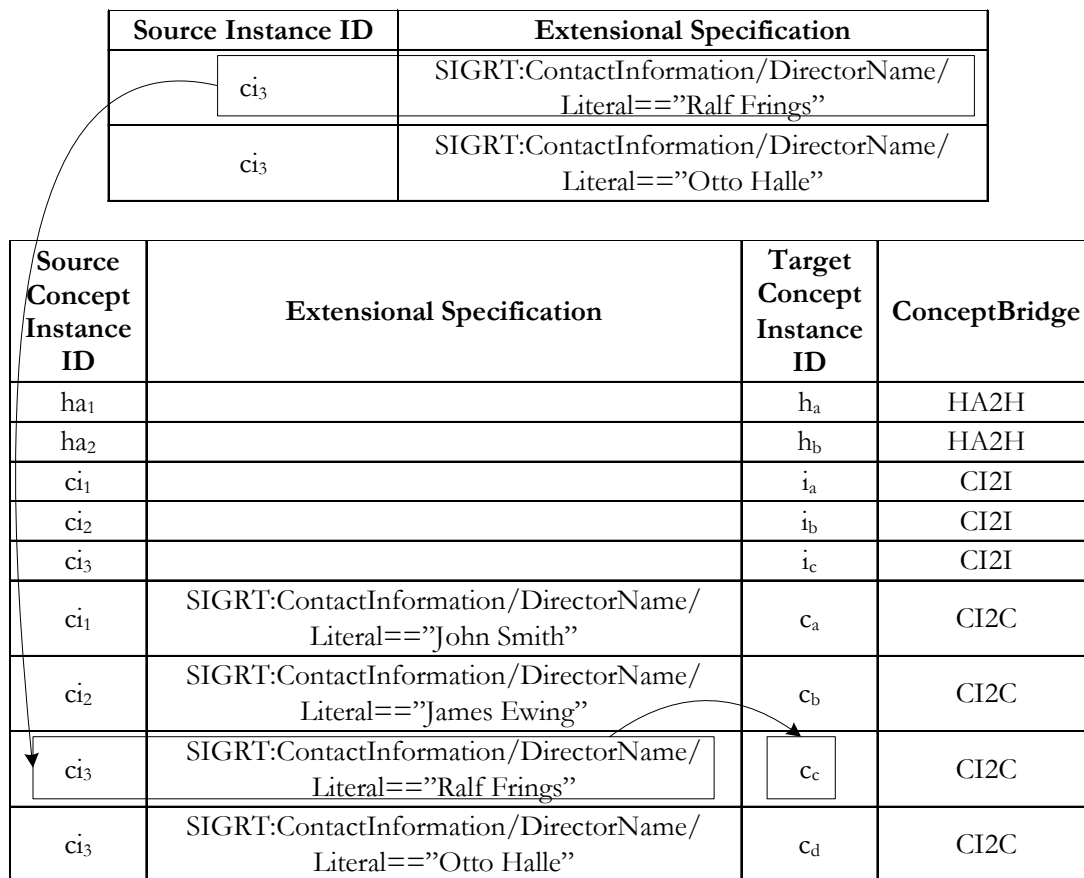


Figure 6.16 – Univocal correlation through extensional specification

Notice that only one line of TI^2 table matches each of the rows of previous table, providing therefore a univocal correlation between source instance perspective and target instance. Accordingly, the TIF:Identification/contact/Contact is instantiated as follows:

$$\begin{aligned} \mathcal{I}_{TIF} &= \{h_a, h_b, i_a, i_b, i_c, c_a, c_b, c_c, c_d\} \\ inst_{\mathcal{C}_{TIF}} &= \left\{ \begin{array}{l} Hotel(h_a), Hotel(h_b), \\ Identification(i_a), Identification(i_b), Identification(i_c), \\ Contact(c_a), Contact(c_b), Contact(c_c), Contact(c_d) \end{array} \right\} \\ inst_{\mathcal{P}_{TIF}} &= \left\{ \begin{array}{l} identification(h_a, i_a), identification(h_a, i_b), identification(h_b, i_c), \\ contact(i_a, c_a), contact(i_b, c_b), contact(i_c, c_c), contact(i_c, c_d) \end{array} \right\} \end{aligned}$$

6.4 Constraints upon target instances

Complementarily to the constraints defined upon the source knowledge base, it is often necessary to constrain the execution process upon the generated target instances.

While this problem relates with any type of constraint, cardinality dimension has been selected to describe and analyze the problem because it is probably the most common constraint upon the target instances.

6.4.1 Analysis of the problem

Sometimes, even if multiple target instances can be created from the same source instance, it is important to define the minimum, maximum or exact number of target instances to create.

The number of target instances to create in the scope of a SemanticBridge depends on the number of existent source instances to transform (i.e. the number FQ table rows), but specially and ultimately on the transformation occurring for every row, which might result in the creation of more than one target instance from each row.

Example 6.25 – Service-dependent cardinality

Imagine that the Service (say SplitRegExpression) intends to split an attribute instance (String) into string tokens according to a string pattern (regular expression). For each resulting token a target instance attribute is created. Multiple target instances can be created from each row.

Therefore, it makes no sense to check the SemanticBridge target cardinality according to the FQ table. Considering previous example, it makes no sense to create some instances of the target attribute if the number of resulting tokens is less than the specified cardinality.

Similar observations occur for any other type of comparison done upon the transformation performed *a posteriori* by the Services.

Another problem relates to the ambiguity of the comparison operators. In particular referring to the cardinality dimension, three cardinality operators have been originally suggested:

Table 6.34 – Cardinality operators constraining the execution of SemanticBridges

Operator	Meaning (true if)
EQCardinality	cardinality of Op1 is equal to Op2
LTCardinality	cardinality of Op1 is less than Op2
GTCardinality	cardinality of Op1 is greater than Op2

These operators compare the number of different instances of Op1 property in *FQ* table with Op2. If the comparison evaluates to *true* it means that the SemanticBridge should be executed, i.e. one target instance is created from each *FQ* table row. Therefore, these comparisons test the executability of the SemanticBridge, but do not states any constraint about the number of target instances to create.

While previous operators could be overloaded to constrain the creation of target instance cardinality, adopting such approach would conduct to ambiguity of the operator and poor semantic expressiveness. In particular, multiple contradictory interpretations would be plausible for several comparison situations (Table 6.35):

Table 6.35 – Ambiguous interpretations of cardinality operators concerning executability and target instance creation

Operator	Comparison operands	Target cardinality interpretations			
		Do not execute	Do not create instances	Create all possible instances	Create instances but
EQCardinality	Op1<Op2	Yes	Yes	-	-
	Op1==Op2	-	-	Yes	-
	Op1>Op2	Yes	-	-	= Op2
GTCardinality	Op1<Op2	Yes	Yes	Yes	-
	Op1==Op2	Yes	Yes	Yes	-
	Op1>Op2	-	-	Yes	-
LTCardinality	Op1<Op2	-	-	Yes	-
	Op1==Op2	Yes	Yes	-	< Op2-1
	Op1>Op2	Yes	Yes	-	< Op2-1

For example, notice that when EQCardinality is used and Op1>Op2, two mutually exclusive interpretations are admissible:

- The constraint is applied to check the cardinality, then the SemanticBridge should not be further executed;
- The constraint states the cardinality of the target instances, then the target instances are created, but only Op2 number of instances.

Resuming, two problems arise from the adoption of constraints upon target instances:

- The evaluation of the target instances according to ConditionExpressions;
- Ambiguity of the comparison operators, originally suggested as comparison operators upon source instances only.

6.4.2 Developed solution

To overcome the first problem, the proposed solution suggests the inclusion of two more phases in the execution process, resulting in the following five-phase process:

1. Querying the source knowledge base (as described previously);
2. Filter the knowledge base query according to the ConditionExpression defined upon the source knowledge base (this phase results in the first *FQ* table);
3. Transformation of the source KB into target ontology instances as described previously, but the resulting instances are reserved instead of the immediate inclusion in the target KB;
4. Filter the target ontology instances resulting from previous phase according to the ConditionExpression defined upon the target knowledge base (this phase results in the second and last *FQ* table);
5. Create the target instances resulting from the last *FQ* table, in the target KB.

As noticed, while the analysis of the problem has been done upon the cardinality dimension, the proposed solution is not focused in this dimension only, but is generic enough to address any dimension and its comparison operators.

With respect to the second problem and in order to improve clarity and semantic expressiveness, it is suggested the specification of new comparison operators upon target knowledge base.

Thus, two groups of cardinality operators have been defined:

- Source cardinality operators, referring to the previously defined EQCardinality, LTCardinality and GTCardinality operator, which check the executability of the SemanticBridge according to the cardinality of the source instances;
- Target cardinality operators, that control cardinality of the target instance creation.

Accordingly, three new target cardinality operators are defined with the following purposes:

Table 6.36 – Target instance creation cardinality operators

Operator	Purpose
Target Entity ExactCardinality Value	Creates exactly Value instances of TargetEntity
Target Entity MaxCardinality Value	Creates at maximum Value instances of TargetEntity
Target Entity MinCardinality Value	Creates at minimum Value instances of TargetEntity

Every target condition expression cardinality expression is evaluated and interpreted accordingly. Each target cardinality expression may evaluate to three distinct results, conducting to distinct interpretations and actions. Table 6.37 provides the interpretation and correspondent action for every result of the operators (Op1 corresponds to cardinality of the Target Entity and Op2 to Value):

Table 6.37 – Cardinality operators constraining the creation of target instances

Operator	Cardinality Comparison result	Interpretations		
		No instance created	Create all possible instances	Execute only Op2 instances
ExactCardinality	Op1<Op2	Yes	-	-
	Op1==Op2	-	Yes	-
	Op1>Op2	-	-	Yes
MaxCardinality	Op1<Op2	-	Yes	-
	Op1==Op2	-	Yes	-
	Op1>Op2	-	-	Yes
MinCardinality	Op1<Op2	Yes	-	-
	Op1==Op2	-	Yes	-
	Op1>Op2	-	Yes	-

Target cardinality expressions are specified through ConditionExpressions, which in turn are Boolean expressions. However, because Or logical expressions permit that more than one of the operands evaluate to true, it is possible to define ambiguous target cardinality expressions.

Example 6.26 – Or-based ambiguous cardinality constraints

Considers that the following target instances have been evaluated for $Path_1$ target entity:

$$Path_1 = \{instance_1, instance_2\}$$

Considers that the following target cardinality expression is evaluated upon previous set:

$$K_4 = \{Or(Path_1 MinCardinality 2, Path_1 MaxCardinality 1)\}$$

Notice that both target cardinality expressions evaluate to true. Yet they are mutually contradictory:

- If the MinCardinality constraint is respected, the MaxCardinality is neglected;
- If MaxCardinality constraint is respected, the MinCardinality is neglected.

In that sense, it has been decided that Or logical expressions cannot be defined in target ConditionExpressions. Additionally, target ConditionExpressions can only be based on properties instances that are presented in the last FQ table.

6.4.3 Example 6.27 - Cardinality annotated example

Consider the ontology mapping scenario between TourinFrance and SIGRT ontologies. Despite nothing is stated in SIGRT ontology, the domain expert determined that Cardinality of `contactInformation` property between `SIGRT:HotelAccommodation` and `SIGRT>ContactInformation` is 1.

In order to ensure the creation of exactly one `SIGRT>ContactInformation` instance, the `I2CI` `ConceptBridge` should be refined. Because it makes no sense to create a `SIGRT>ContactInformation` instance if it is not related with a `SIGRT:HotelAccommodation` instance, the `I2CI` `ConceptBridge` should be defined between `TIF:Hotel` and `SIGRT>ContactInformation` for one and only one instance of identification relation. This description is fully characterized by the following specification:

$$I2CI = \left(\begin{array}{l} TIF : Hotel, SIGRT : ContactInformation, \\ \{ \}, \{ \}, \{ \}, \mathcal{K}_{I2CI}, \mathcal{X}_{I2CI}, \mathcal{X}_{I2CI} \\ \mathcal{K}_{I2CI} = \{ C_{I2CI.1} \} \\ C_{I2CI.1} = W_{ST.1}^s \text{ ExactCardinality } 1 \\ \mathcal{X}_{I2CI} = \{ C_{I2CI.2} \} \\ C_{I2CI.2} = W_{ST.1}^s = W_{ST.1}^s \\ \mathcal{X}_{I2CI} = \{ (C_{I2CI.2}, \text{extensionalSpecification}) \} \end{array} \right)$$

$$W_{ST.1}^s = TIF : Hotel / \text{identification} / \text{Identification}$$

The execution of this `ConceptBridge` for the h_1 `TIF:Hotel` instance will give rise to the FQ table represented in Table 6.38.

Table 6.38 - FQ table evaluated for `I2CI` `ConceptBridge` in the scope of h_1 instance

Hotel/identification/Identification
i_1
i_2

Because all execution constraints (from Table 6.37) succeed (i.e. it is possible to create exactly one target instance), the `ConceptBridge` will give rise to one `SIGRT>ContactInformation` instance. In that sense the resulting TI^2 table is (Table 6.39):

Table 6.39 - TI^2 table resulting from the refined $\mathcal{M}_{TIF-SIGRT}$ execution

Source Concept Instance ID	Extensional Specification	Target Concept Instance ID	ConceptBridge
h_1		ha_1	H2HA

h ₂		ha ₂	H2HA
h ₁	Hotel/identification/Identification=i ₁	ci ₁	I2CI
h ₂	Hotel/identification/Identification=i ₃	ci ₂	I2CI

Because an extensional specification has been used to create SIGRT:ContactInformation instances, it becomes necessary to refine the i2ci PropertyBridge such the same target cardinality expressions and extensional specifications are defined as for the I2CI ConceptBridge.

$$i2ci = (\mathcal{W}_{TS.i2ci}^s, \mathcal{W}_{TS.i2ci}^t, \{ \}, CopyRelation, \phi_{TS.i2ci}^s, \phi_{TS.i2ci}^t, \{ \}, K_{I2CI}, \mathcal{X}_{I2CI}, \mathcal{X}_{I2CI})$$

Table 6.40 despite the extensional specification perspectives calculated by the i2ci PropertyBridge in the scope of the ha₁ instance:

Table 6.40 – Different order of the same source instance perspectives

Source Instance ID	Extensional Specification
h ₁	TIF:Hotel/identification/identification=i ₂
h ₁	TIF:Hotel/identification/identification=i ₁

Notice that while the achieved rows are identical to those of Table 6.39 their order is different. The execution engine would try to create a relationship between ha₁ and the target instance originated from the source instance perspective defined according to first row of the table. However, because such instance does not exist the engine will provoke an exception. If the process continues for the next source instance perspective instead, it will succeed in creating the inter-relation, because such perspective indeed originated a target instance.

In fact, because order in query tables is unpredictable, when generating the same source instance perspectives in the scope of different SemanticBridge the order might be different.

As observed, the fact that one of the source instance perspectives does not match TI^2 entries does not mean the execution must fail, but that another source instance perspective should be used. Therefore, in case one source instance perspective fails the match, the process continues to the next perspective until the cardinality has been reached or no further source instance perspectives are available. CopyRelation Service must be therefore modified in a way exceptions are raised only if both next conditions are verified:

- All source instance perspectives have been matched against TI^2 rows;
- The target cardinality has not been reached (after previous condition).

Yet, this solution does not exclude the need to specify, in the PropertyBridge, a target cardinality smaller or equal to the target cardinality specified in the ConceptBridge that created the target instances the PropertyBridge is inter-relating.

6.5 Conclusions

The execution process described in this chapter is based on the instantiation of the SBO ontology into a meaningful mapping specification. Section 6.1 described the fundamental principles of the execution process, namely the execution order between ConceptBridges and PropertyBridges and the Transformation Information Table (TI^2).

One of the most important contributions of this thesis has been presented in Section 6.2, and is referred as the internal process. The core stage of the internal process is the process of querying the source knowledge base. In this stage a new tree-based representation of Paths has been developed, which permits to query and combine instances of the Paths coherently, even when they share some parts of the branches. The process has been described according to relational algebra, providing a well-established ground for the required analysis and argumentation of the problem, as well as the tools to propose and deploy solutions to the problem.

Due to modeling decisions, the proposed SBO does not provide support to property to concept semantic relations. This limitation has been described and a solution proposed based on the extensional specification mechanism. Yet, the solution caused no fundamental changes in SBO, which maintained its core philosophy, structure and semantics. The developed extensional specification mechanism provides the missing features by forwarding the required changes to both the CopyInstance and CopyRelation Services, while maintaining the fundamentals of SBO and of the execution process. These two Services evolved during Section 6.2.4 both concerning the semantics and interface, but the execution process cycle as well as the internal process did not change. What is more, SBO entities and developed internal mechanisms have been successfully applied to the new extensional specification mechanism with minimal or no changes.

Section 6.4 concerned with the target execution process constraints defined upon the target instances. The execution process previously defined does not provide enough features to deal with this requirement. Additionally, it has been noticed that the proposed operators of SBO could lead to ambiguity in the semantic bridging and execution processes. In respect to the first problem, it has been proposed the extension of the three-phase query-filtering-transformation process, into the five-phase query-filtering-transformation-filtering-instantiation process. This process filters the source and target instances in different moments of the process, constraining the flow of process when required and possible. In respect to the problem raised by the ambiguity of the operators, three new cardinality operators have been proposed and their semantics defined, which suggest the separation between source and target cardinality operators. Distinguishing between source and target comparison operators might be adopted for other operators, but this should be decided in a case by case basis.

It is now important to compare the requirements identified in Chapter 5 with the functionalities added in this chapter and supported by the execution process (Table 5.6).

Table 6.41 – Semantic relations characteristics supported by SBO and execution process

Dimensions		Semantic Relations Characteristics	Required support	SBO support	Added support
1. Entity type		Concept to Concept	Yes	Full	-
		Concept to Property	Yes	Full	-
		Property to Concept	Yes	No	Full
		Property to Property	Yes	Full	-
		Entity to Instance	Limited	No	-
2. Transformation	2.1 Function	Set of basic functions	Yes	Full	-
		Combination of functions	Yes	No	-
		Integration of new functions	Yes	Full	-
	2.2 Directionality	Unidirectional	Yes	Yes	-
		Manually bidirectional	Limited	No	-
		Automatically Bidirectional	Limited	No	-
3. Cardinality		0:1	Yes	n/a	-
		0:n	n 0:1 bridges	n/a	-
		1:1	Yes	n/a	-
		1:n	Yes	n/a	-
		n:0	Limited	n/a	-
		n:1	Yes	n/a	-
		m:n	by m:1+1:n	n/a	-
4. Constraint		Not constrained	Yes	Full	target instance cardinality
		Ontological entities-based	Yes	Yes	
		Non-ontological entities-based	Yes	Yes	
5. Structural support		Object-oriented	Yes	Yes	-
		Property-centric	Yes	Yes	-
		Flow execution control	Yes	Yes	-

Previous table illustrates the contribution of the execution process to the fulfillment of the support envisaged to the ontology mapping system. However, probably more important than the support added by the execution process, it is the fact that all features of semantic bridging phase supported by SBO are supported by the execution process. The exception is the Entity to Instance semantic relation that is not yet supported by the execution process.

Therefore, the main research contributions of this chapter are:

1. The generic execution process together with the transformation information table that relates each created target concept instance with the source concept instance that originate it;
2. The internal process described in terms of relation algebra, where source knowledge base is semantically queried and filtered according to SemanticBridges arguments;

3. Extensional specification of non-explicitly existent source concept instances, which permits to support property to concept semantic relations, maintaining SBO fundamental characteristics, namely the clear and versatile set of inter-relations between ConceptBridges and between ConceptBridges and PropertyBridges;
4. The developed solution to control target instances cardinality based on generic ConditionExpressions.

Chapter 7

MULTI-DIMENSIONAL

SERVICE-ORIENTED ARCHITECTURE

This chapter describes the research done in the analysis and specification of the architecture of the ontology mapping system according to the research proposals suggested in previous chapters. The work described in this chapter has been initially published in [Silva *et al.*, 2003; Silva & Rocha, 2003c; Silva & Rocha, 2003e; Silva & Rocha, 2004a; Silva & Rocha, 2004b].

The major goal of the architecture is not only to propose a model for the implementation of the ontology mapping system, but to apply, respect and exploit the ideas explicitly or implicitly represented through the MAFRA – MAPPING FRAMework, and especially the ideas proposed in the semantic bridging and execution phases.

In fact, the proposed architecture is the result of the combination of many different valuable knowledge inputs, and it represents one of the major outcomes of the research work described in this thesis.

First section of this chapter analyzes the problem of the automation of the ontology mapping system. Second section presents the proposed architecture, describing its components and inter-relations. The third chapter describes the research efforts made in the specification of the automatic semantic bridging process, as a case test of the proposed architecture.

7.1 Observations

Because the ontology mapping process is a time-consuming, knowledge-demanding, user-based process, automation is a very important feature for most of the application scenarios suggested in Chapter 2.

However, automation is difficult and error-prone, especially due to the incompleteness and subjectivity of ontologies. In fact, despite the fact ontology is a formal and explicit specification of a conceptualization, it reflects the subjectivity of the conceptualization and might therefore be difficult to understand and manage. Because of a subjective element, some of the phases of the process are inherently subjective, and therefore hard to automate.

This is especially true for the semantic bridging and similarity measuring phases, but the cooperative consensus building and evolution tasks also suffers its effects. Next three sections describe the observed requirements and limitation in automating these tasks.

7.1.1 The gap between similarity measuring and semantic bridging

Current ontology mapping systems [Dou *et al.*, 2002; Omelayenko, 2002a; Omelayenko, 2002b; Stuckenschmidt & Visser, 2000; Stuckenschmidt & Wache, 2000; Xiao *et al.*, 2004] are capable to represent and execute semantic relations defined by the domain expert, but none is capable to support the automatic definition of semantic relations. On the other hand, many research efforts [Beneventano *et al.*, 2001; Dionísio *et al.*, 2001; Doan *et al.*, 2002; Kang & Naughton, 2003; Miller *et al.*, 2000; Resnik, 1999] is running on developing similarity measuring systems, the so called matchers, whereby similarities between ontologies entities are discovered and measured. Distinct approaches are adopted, such as linguistic disambiguation [Dionísio *et al.*, 2001; Resnik, 1999], clustering [Beneventano *et al.*, 2001; Doan *et al.*, 2002], graph-based [Beneventano *et al.*, 2001; Miller *et al.*, 2000] and instance-based [Kang & Naughton, 2003] analyses. Matchers are, in most cases, capable to determine the existence of a semantic relation between a pair of ontologies entities [Doan *et al.*, 2002; Kang & Naughton, 2003; Miller *et al.*, 2000], and in some cases are even able to

determine the existence of 1:n semantic relation [Beneventano *et al.*, 2001]. However, no matcher is capable to determine the type of transformation occurring in the semantic relation. Resuming:

- Matching is capable to discover and measure the existence of semantic relations;
- Ontology mapping system is capable of representing the semantic relations;
- Neither matching nor mapping processes are capable to determine the type of transformation associated with the semantic relation.

An import gap exists therefore between ontology mapping systems and similarity measuring systems, consisting in determining the correct Service to associate with a SemanticBridge according to a set of similarity measures. Overcome this gap is one of the major goals of any ontology mapping system, which corresponds to the automation of the semantic bridging phase.

User-based specification of SemanticBridges, including the application of a Service to every SemanticBridge, is based on heuristics rules constrained by the set of related entities and the transformation performed by the Service. The primary approach consists therefore in capturing these heuristic rules into a rule-based system, which would be applied in deciding the suitability of the Service in context of each SemanticBridge.

However, due to the dynamics and subjective nature of the ontology mapping scenarios, it is natural that new transformation capabilities would be often required, motivating the appearance of new Services. In turn, new Service development results in changes in the rule-based system, including the addition (and eventually modification) of new heuristic rules according to the new Services requirements. This evolution requires considerable expertise upon the rule-based system by the Service developer which otherwise could generate undesired effects in the already defined Services.

7.1.2 Cooperative consensus building

A new dimension is added to the ontology mapping process when, instead of a unique actor (user), two or more actors take part and influence the ontology mapping process. In fact, within a context of multiple actors, there is not anymore a unique subjective interpretation and decision, but instead multiple subjective interpretations trying to influence the final decision.

The semantic bridging phase is envisaged as the one profiting more from the cooperative consensus building capabilities of the system, and is therefore the special focus of this observation.

Cooperative consensus building at semantic bridging phase aims to either:

- Achieve a consensus between two entities mapping the ontologies;
- Improve the quality of the mapping by recruiting third-party entities competencies into the final mapping document;

- Provide the mechanisms to support user in deciding about two or more (eventually exclusive) SemanticBridges.

The goal however, is not to derive SemanticBridges but to combine the perspectives of the intervenients about a set of SemanticBridges and derive a mapping document with certain characteristics (e.g. better quality, commonly accepted). In fact, the cooperative consensus building process starts when the different intervenients already have their own proposals about the SemanticBridges. This characteristic clearly distinguishes the semantic bridging process from the cooperative consensus building process. Consequently, the process will naturally focus on negotiating the characteristics of SemanticBridges suggested by the intervenients, which in turn results in arguing about the competency and semantic validity of applying a specific Service in relating a set of source and target entities. Services arise once again as a corner stone in the process.

7.1.3 Evolution

Once the mapping document is defined and “running”, it often occurs that mapped ontologies evolve by a multitude of reasons. In these circumstances, the ontology mapping specification often becomes schematic or semantically incorrect. Schematic errors occur when the ontology mapping is no longer compatible with the schematic elements of the ontologies.

Example 7.1 - Schematic evolution of ontologies demand SemanticBridges evolution

Consider that the PropertyBridge price2price exists between O1:Item.price.Literal and O2:Article.price.Literal, in which CopyAttribute is the applied Service. Meanwhile O1 ontology evolved such property O1:Item.price.Literal has been substituted by the O1:Person.priceEUR.Literal property. The price2price PropertyBridge becomes schematic invalid because the applied source property no longer exists in the ontology.

Semantic errors occur when the mapped ontologies change in a way that semantic relations in the mapping document are no longer semantically valid.

Example 7.2 - Semantic evolution of ontologies demand SemanticBridges evolution

Consider the initial price2price PropertyBridge of previous example. Also, consider that O1 ontology evolved such the O1:Item.price.Literal property no longer represents the price in PTE currency but in EUR currency. As consequence, it is no longer semantically correct to copy instances (transformation provided by the CopyAttribute Service) of O1:Item.price.Literal to O2:Article.price.Literal since both properties represents different values. However, the price2price PropertyBridge is still schematic correct.

Managing evolution of the ontology mapping document is, in some aspects, similar to the semantic bridging process, but there are a few important differences:

- Parts of the mapping document are eventually not affected by the ontologies changes, and should therefore be kept unchanged;

- Even the affected parts of the existent mapping document contains valuable, eventually correct semantic bridging information, reflecting important user-based decisions that can/should be exploited in the evolution process.

Thus, the evolution process of mapping should respect not only the ontologies but also the already defined SemanticBridges. Existent SemanticBridges and their associated Services arise consequently as one of the main reasoning elements to consider in the evolution task.

7.1.4 Synthesis

According to previous descriptions and considering all knowledge described in previous chapters, two important facts are systematized:

1. The central role of Services. Service, as one of the core components of the proposed ontology mapping approach, is responsible for the transformation capabilities of the ontology mapping system. Because every SemanticBridge has an associated Service, the types of SemanticBridges and thus of the transformation capabilities of the ontology mapping system are ultimately dependent on the transformation Services available in the system.
2. The virtually incomplete transformation capabilities. As referred along this thesis, the transformation requirements vary according to the heterogeneity of the ontologies and their subjective nature, and the user interpretation of the semantic relations holding between them. In that sense it is virtually impossible to provide transformation capabilities, by the ontology mapping system, capable to solve any mapping scenario.

7.2 Proposal

This section describes the developed approach in order to address the problems raised by the automation of the different phases of the ontology mapping process, resulting in the so called Multi-Dimensional Service-Oriented Architecture.

Previous observations together with the seven quality vectors introduced in 4.1, lead to the adoption of three distinct but interrelated lines of research:

1. Improve the competency of Services so they can contribute with specific know-how to other phases of the overall mapping process that would request their competencies and know-how. Services are no longer limited to transform source instances into target instances during the execution process, but are endowed with multiple types of competencies, arising as the so called Multi-Dimensional Services;
2. Turn Services independent from any core MAFRA module, and not intrinsically belonging to the execution module. The relation between Services and core modules evolves from a dependent, subservient relation of Services in relation to the modules, to a cooperation-based

relation. Competencies (know-how) are requested by modules and provided by Services, clearly separating both types of entities. Services and modules should therefore cooperate according to a commonly established interface;

3. Model Services as pluggable, self-describing entities, supporting the capability of the ontology mapping system to adapt to new mapping scenarios, while respecting and promoting the independence of Services from core modules. This arises from, and drives to, a generalization of the previous line of attack, in which Services are made so independent from the core system that they can be made external to the system. Services should therefore provide their own description of competencies and requirements in a way the core system can recruit their competencies according to each phase requirements.

The result is the so called Multi-Dimension Service-Oriented Architecture, illustrated in Figure 7.1.

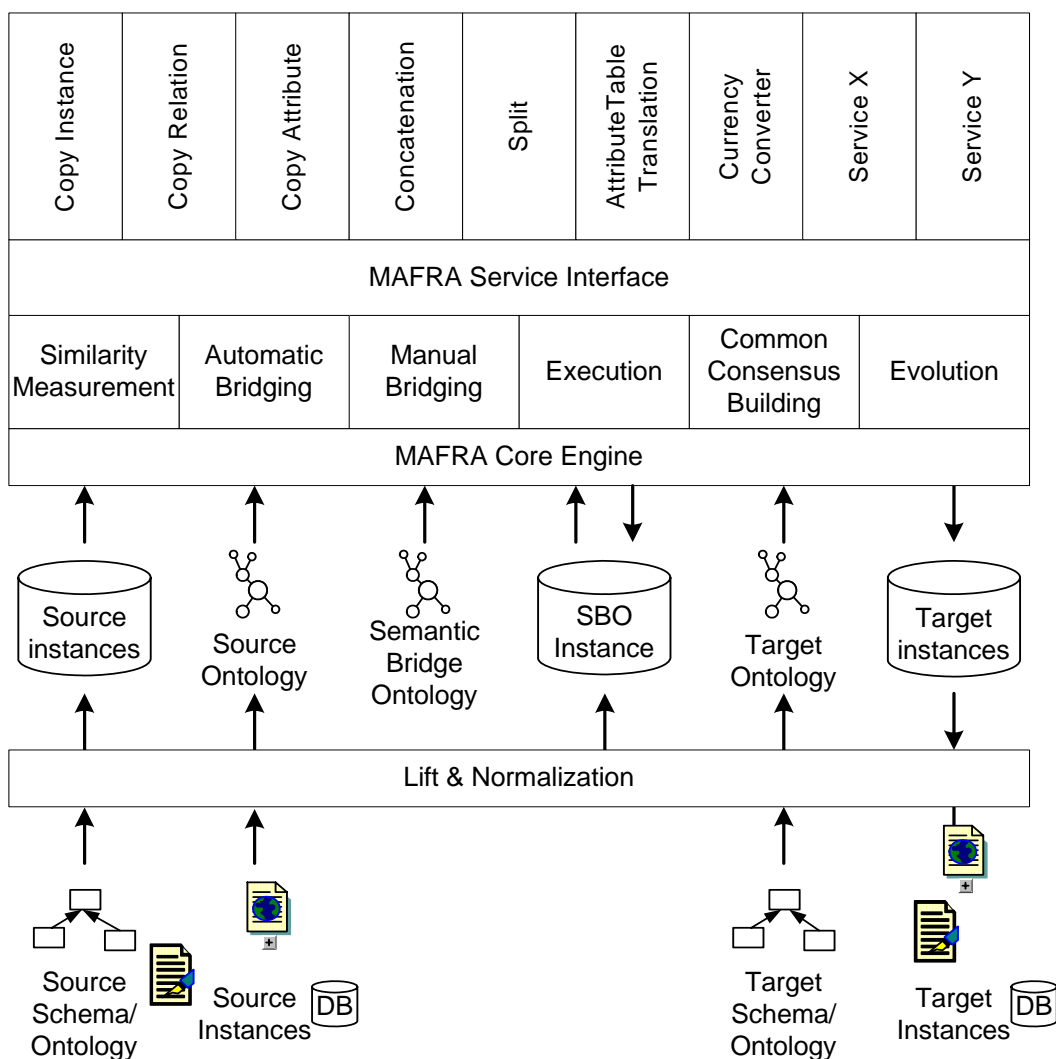


Figure 7.1 – Multi-Dimensional Service-Oriented Architecture

This architecture reflects the adoption of a modular, decentralized architecture, where Services are attached to the system functional core modules (i.e. bridging, execution, common consensus building, evolution, etc.) through a specific interface, referred as the MAFRA Service Interface. Services are therefore understood as independent, dynamic, intelligent entities, encompassing the better as possible the user-acquirable and user-acquaintable know-how, providing different competencies as required by each core modules. The MAFRA Core Engine is responsible for general tasks, such as loading and storing ontologies and knowledge bases, and processing and dispatching user-requests to core modules.

7.3 Automatic Semantic Bridging

In order to demonstrate the application and exploitation of the Multi-Dimensional Service-Oriented Architecture, the automation of the semantic bridging phase will serve as a case test. The goal of the process is to choose the right combinations of source entities with target entities through a Service.

7.3.1 Observations

Recurring to lexical tools like dictionaries, specific domain thesaurus and WordNet, it is possible to classify multiple similarities between pair of entities. However, these classifications are typically insufficient and error-prone due to the classification defined by the annotated corpus used in the process. Structural analysis of ontological entities and statistical and probabilistic analyses of ontologies instances are other ambiguous or insufficiently accurate similarity measuring techniques. In fact, according to literature [Bernstein & Rahm, 2001; Doan *et al.*, 2002; Rahm & Bernstein, 2001] and analyzing their respective results, it is empirically evident that no technique or algorithm is accurate enough, thus the need to combine efforts from all knowledge sources possible into a useful system.

7.3.2 Hypothesis

Instead of research on new, eventually better similarity systems capable to assign a Service to a SemanticBridge, it is suggested to research on the technology that permits the semantic bridging phase to adopt and exploit the similarity measuring systems already available, through the independent and specific know-how of Services.

It is proposed to adopt and exploit the evaluated similarity measures according to each Service specificities and specifications, so the system is able to decide the best Service to apply between two sets of ontologies entities. However, instead of centering that competency in the automatic semantic bridging core module, the Multi-Dimensional Service-Oriented Architecture is exploited.

For that, each Service capabilities are enhanced in a way that, through the set of provided similarity measures, it can reason, emit an opinion or decide about its aptitude to relate a set of source ontology entities with a set of target ontology entities.

In order to maintain the system dynamic and as open as possible, each similarity measuring system would be implemented by an external, pluggable entity connected to the core system by a common specific interface (just as described for Services).

7.3.3 Specification

This section describes the systematization and specification process of the proposed approach. In the first sub-section the fundamental input entities and objects of the system are described: matchers and matches respectively. Second sub-section describes the core automatic bridging process, referred as Clustering. It consists in grouping matches into meaningful sets so a specific Service is able and competent to transform the instances of the ontologies entities referred in matches. As central process in the system, clustering is the ultimate responsible for the quality of the resulting SemanticBridges. Yet, clustering is not itself a specific monolithic piece of the system but is instead distributed by Services. Thus changes or refining clustering is then preferably performed incrementally and according to each Service specificities.

7.3.3.1 Matchers and Matches

Similarity measuring is the process performed by the similarity measuring systems, referred as matchers entities, which evaluate the similarities between pairs of source and target ontologies entities. The similarity measure is referred as a match and denotes a certain degree of a certain type of similarity between the source and the target ontology entities.

Example 7.3 – Matchers and the different dimensions of ontologies

One matcher might measure the similarity between two entities according to their names, while another matcher might use the lexical information associated with the entities, while another might use the hierarchical structure in which they are defined.

Therefore, in the context of this thesis, a match is a structure in the form of:

$$Match := \left\langle \begin{array}{l} Source\ ontology\ entity, Target\ ontology\ entity, \\ Matcher, Similarity\ value, Justifications \end{array} \right\rangle$$

where:

- *Source ontology entity* is the source ontology concept or property addressed in the match;
- *Target ontology entity* is the target ontology concept or property addressed in the match;

- *Matcher* is the identification of the matcher that evaluated the similarity measure, which in turn characterizes the type of similarity measured. At maximum one match is evaluated by a matcher for the same pair of ontology entities;
- *Similarity value* corresponds to the degree of the similarity, normally ranging from 0 to 1;
- *Justifications* is the set of statements providing argumentation for the evaluated value. Currently, justifications are not being used, but their application is envisaged in the common consensus building part of the process.

Because multiple matchers are supposedly available in the system, multiple matches may be evaluated for the same entity (either source or target ontology entity) potentially forming an n:m relation. It is therefore beneficial to index the repository of matches (M) by (i) the source entity, (ii) the target entity and (iii) the matcher. Moreover, the set of all matches evaluated for the same pair of ontology entities is referred to as a Knot.

Example 7.4 – Similarity measuring scenario

Consider the ontology mapping scenario of Figure 7.2 where ontology O1 is to be semantically related to ontology O2.

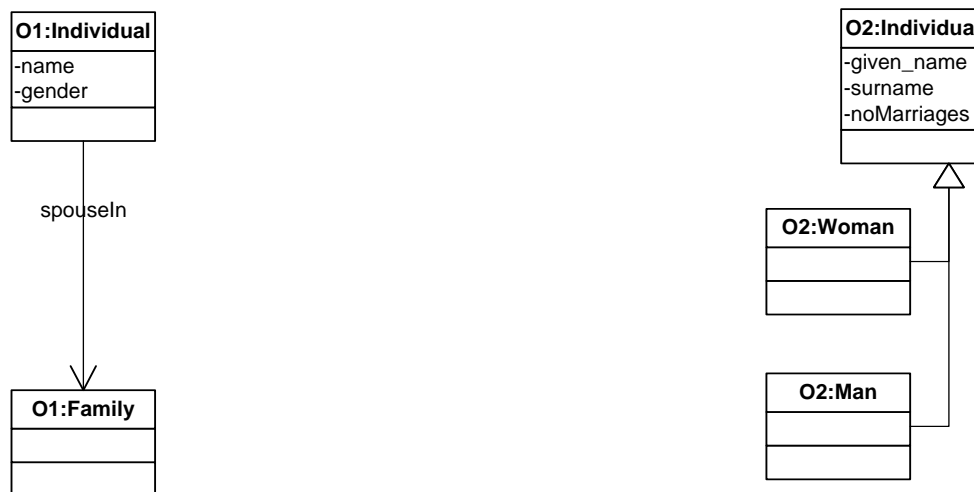


Figure 7.2 – Simple ontology mapping scenario using UML notation

Consider also the three available matchers:

- Resnik-like matcher, based on the Resnik method [Resnik, 1999] on word-sense disambiguation (WSD);
- MOMIS-like matcher, based on structural and clustering techniques suggested on MOMIS [Bergamaschi *et al.*, 1999];
- Type-checker matcher, which is capable to determine the types of ontology entities, either by the information in the ontology or by instance-based analysis.

Table 7.1 represents the matches resulted from the similarity-measuring phase whose value is above certain predefined global threshold.

m_5 and m_8 compose the knot for the O1: name and O2:surname ontology entities.

Table 7.1 – Example of matches resulting from the similarity measuring phase

ID	Source entity	Target entity	Matcher	Similarity value	Justifications
m ₁	Individual	Individual	Resnik-like	0.95	∅
m ₂	Individual	Man	Resnik-like	0.86	∅
m ₃	Individual	Woman	Resnik-like	0.86	∅
m ₄	name	given_name	Resnik-like	0.82	∅
m ₅	name	surname	Resnik-like	0.82	∅
m ₆	spouseIn	noMarriages	Resnik-like	0.66	∅
m ₇	name	given_name	MOMIS-like	0.81	∅
m ₈	name	surname	MOMIS-like	0.85	∅
m ₉	Individual	Individual	MOMIS-like	0.78	∅
m ₁₀	Individual	Man	MOMIS-like	0.78	∅
m ₁₁	Individual	Woman	MOMIS-like	0.78	∅

Notice that at similarity measuring stage, ontology entities are treated independently of their type, permitting matches between concepts and properties and vice-versa. This situation is not illustrated in previous example.

In addition to the matches proposed by (automatic) matchers, other entities in the system, like the domain expert, can provide their own matches. The matcher identification in such matches is set according to the identity of the entity. For instance, the matcher element in the matches defined by the domain-expert (the user) will be fulfilled with “user”. Other entities will propose matchers, as it will be referred in next sections.

7.3.3.2 Cluster and Clustering

Clustering is the process that groups together a set of matches into an entity named Cluster. The source entities defined in these matches are semantically related to the target entities defined in the same matches. The output of the clustering process is a set of clusters, each being characterized according to the type of related entities and its cardinality. Cluster cardinality refers to the number of distinct source ontology entities and the number of distinct target ontology entities correlated in the cluster.

Example 7.5 – Matches forming a cluster

From the set of matches presented in Table 7.1, a common-sense meaningful cluster is composed by the matches relating “name” to “given_name” and “surname” (Table 7.2). The cardinality of this cluster is 1:2 (1:n generically).

Table 7.2 – Matches forming a cluster

ID	Source entity	Target entity	Matcher	Similarity value	Justifications
m ₄	name	given_name	Resnik-like	0.82	∅
m ₅	name	surname	Resnik-like	0.82	∅
m ₇	name	given_name	MOMIS-like	1	∅
m ₈	name	surname	MOMIS-like	1	∅

MOMIS [Beneventano *et al.*, 2001] and SKAT [Mitra *et al.*, 1999] are the few known systems claiming the capability to group matches together. While SKAT clustering capabilities are rather simple, MOMIS can effectively group together semantically related entities based on the WordNet lexical relations between entities labels. A “global as view” approach is used, in which mapping rules, derived manually by the designer (expert), relates the ontology entities to the global entity. However, no transformation function is explicitly defined between entities, but instead logical operators are used, leaving the semantic relation extremely ambiguous.

Example 7.6 – MOMIS generated cluster

The next piece of code represents a mapping rule derived between ID ontology (Intensive Care Department) and the CD ontology (Cardiology Department).

```
attribute name
  mapping_rule
    (ID.Patient.first_name and ID.Patient.last_name),
    CD.Patient.name
```

According to the nomenclature used during this thesis, the previous mapping rules would correspond to the PropertyBridge between the set of source entities ID:Patient/first_name/Literal and ID:Patient/last_name/Literal, and the target entity CD:Patient/name/Literal. No Service is explicitly stated (the “and” operator is used), Even if for a human it is obvious that the Concatenation Service should be applied instead of “and”, a multitude of other transformations are possible in the scope of the ontology mapping system.

Yet, it is of fundamental importance to ontology mapping system that the transformation function operating between ontology entities instances is explicitly defined instead of other possibilities.

Consequently, Cluster should be further characterized with the Service capable to transform instances in the ontologies entities. A cluster is then a structure in the form of:

$$Cluster := \langle Service, Matches \rangle$$

where:

- *Service* is the name of the Service capable to transform the instances of the ontologies entities defined in *Matches* ;
- *Matches* is a set of matches.

Example 7.7 – Service-Cluster association

Consider that the Split Service would be adequate to process the entities referred in the matches represented in Table 7.2. The resulting cluster (cl) would be represented as:

$$cl = \langle Split, \{m_4, m_5, m_7, m_8\} \rangle$$

7.3.4 Reducing combinatorial space by Service-based clustering

This section describes the research work developed in the scope of this thesis concerning the reduction of the combinatorial space of the semantic bridging process. The described method does not intend to define the correct SemanticBridges, but instead to define all possible SemanticBridges. Applying a simple definition process permits to focus efforts in better judge the suitability of the Service in relating the set of ontologies entities.

The proposed method is based on clustering matches according to Services interface. In special two characteristics are useful:

- Cardinality. Cardinality of the match should match the Service cardinality. In case they do not match, the Service is considered unable to semantically relate the cluster;

Example 7.8 – Clustering constrained by the Service cardinality

It makes no sense to try to associate the CopyAttribute Service to the cluster specified in Table 7.2. In fact, the CopyAttribute Service is a 1:1 cardinality Service, while the cluster is 1:2 (1:n, generically). Instead, the Split Service (1:n cardinality) is not disregarded immediately.

- Types of the arguments, constrain the type of entities semantically related in the cluster.

Example 7.9 – Clustering constrained by the type of arguments of Service

Consider the set of matches presented in Table 7.1. According to m1 and m9 matches, O1:Individual is semantically related to O2:Individual. Because both CopyAttribute and CopyInstance Services have 1:1 cardinality, both could be applied in an eventual SemanticBridge between these two entities. However, the CopyAttribute Service arguments do not support concepts, but attributes. Instead, CopyInstance Service arguments are of type concept. Thus, while the CopyAttribute Service would be considered unable, the CopyInstance Service would prevail as a possibility.

The primary constraint the system must respect to is the Service interface. In fact, each Service is characterized by a number and type of arguments, which must be respected when applied in a specific SemanticBridge. Driving the clustering process according to the Services interface will reduce easily and efficiently the combinatorial space. Such approach is commonly adopted in other disciplines such in software engineering in which the interface of a procedure or function is used to (limitedly) determine the program correction (compilation) and generate the assemble code (linkage).

Despite clustering process is dependent on each Service, a common approach has been systematized respecting the implementation of this method. The derived approach suggests that

every match evaluated by matchers is pushed to every Service in the system, which decides what to do according to its own interface. This situation is metaphorically represented in Figure 7.3, in which a part of the Multi-dimensional Service-oriented Architecture is represented. According to the Services decisions, clusters (e.g. cl_1 , cl_2 , cl_3) are created and associated with the Service that created it.

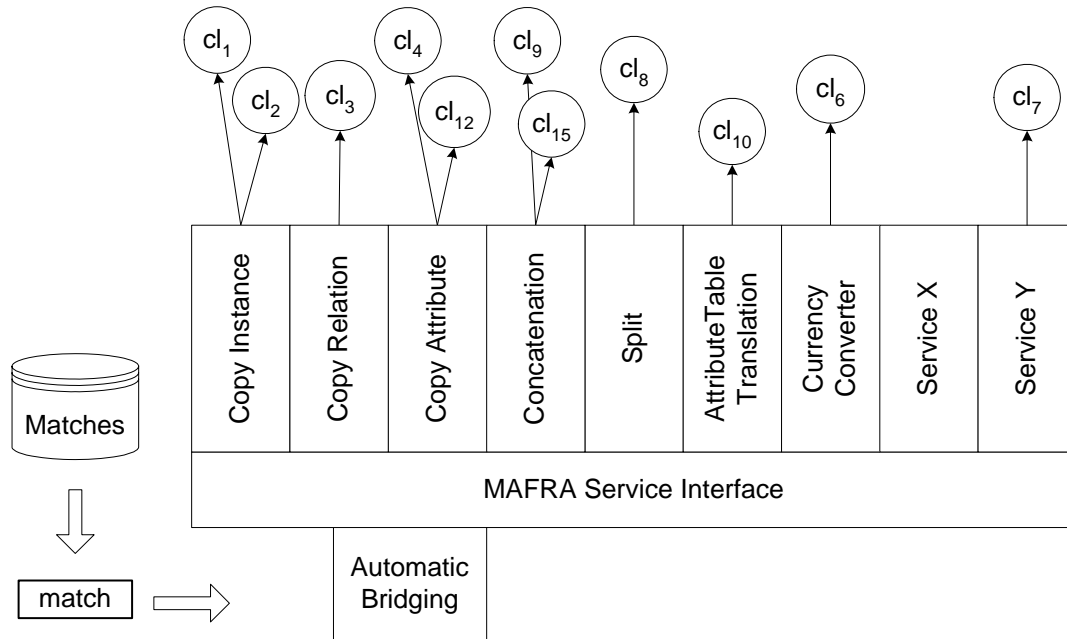


Figure 7.3 – Service-based clustering for reduce combinatorial space

Essentially, the derived approach states that in every Service the match motivates one of the following judgments:

1. The match provokes the creation of a new cluster, in which case a new cluster is created and the Service associated with it;
2. The match makes sense in the scope of an already existent cluster, in which case the match is added to that cluster;
3. The match produces the removal of an existent cluster, in which case the cluster is removed;
4. The match is useless in the scope of the Service and is therefore disregarded.

While conceptually corresponding to these four judgments, the process is implemented slightly different. Basically, the implemented process verifies for every Service if there is a set of matches whose (source and target) entities had no further matches and fit the interface of the Service. As consequence, once a cluster is created, no match can motivate its removal.

Considering a repository of matches M , the process corresponds to the following steps:

1. For every Service available in the system, fill in $M_{Service}$ with all the matches from M whose entities conform the types of the Service arguments;

2. For every Service, subtract⁴⁴ from $M_{Service}$ a match m and the matches from its knot into k_m . Evaluate:

- $M_{Service}^{se}$ as the set of all matches subtracted from $M_{Service}$, in which the source entity of m is referred;
- M^{se} as the set of all matches retrieved⁴⁵ from M , in which the source entity of m is referred and are not present in $M_{Service}^{se}$. Contrary to $M_{Service}^{se}$, M^{se} may contain matches whose types do not conform to the interface;
- $M_{Service}^{te}$ as the set of all matches subtracted from $M_{Service}$, in which the target entity of m is referred;
- M^{te} as the set of all matches retrieved from M , in which the target entity of m is referred and are not present in $M_{Service}^{te}$. Contrary to $M_{Service}^{te}$, M^{te} may contain matches whose type do not conform to the interface;

3. According to the Service cardinality (i.e. 1:1, 1:n and n:1), three situations are possible:

3.1 Cardinality 1:1. If $M_{Service}^{se}$, M^{se} , $M_{Service}^{te}$ and M^{te} are empty, it means that both source and target entities of m are matched together and uniquely. As consequence:

- All matches from k_m give raise to a new cluster and are associated with it;

3.2 Cardinality 1:n. Three situations may occur:

3.2.1 If $M_{Service}^{se}$ is empty it means that 1:n cardinality for the source entity will never be achieved and therefore m should be discarded and proceed to the next match (step 2);

3.2.2 If $M_{Service}^{se}$ is not empty and M^{se} , $M_{Service}^{te}$ M^{te} are empty, it means that source entity of m is related with more than one target entity, which fits the Service interface. However, it is necessary to check if such target entities match with this source entity only. To check this situation the process evaluates $M^{te'}$, which corresponds to all matches subtracted from $M_{Service}$ and retrieved from M whose target entities are referred in matches from $M_{Service}^{se}$. If $M^{te'}$ is empty, it means that no target entity is related to more than the source entity. Therefore:

- All matches from k_m and $M_{Service}^{se}$ give raise to a new cluster and are associated with it;

3.2.3 In any other circumstances an n:m relation exists, which is not supported by the proposed method. Accordingly all matches from $M_{Service}^{se}$ are discarded and the

⁴⁴ In this context, to subtract means to fill in a set by removing the elements from initial set.

⁴⁵ In this context, to retrieve means to fill in a set by copying the elements from the initial set.

process continues to the next match (step 2);

3.3 Cardinality n:1. three situations may occur:

3.3.1 If $M_{Service}^{te}$ is empty, it means that the n:1 cardinality for the target entity will never be achieved and therefore m should be discarded and proceed to the next match (step 2);

3.3.2 If $M_{Service}^{te}$ is not empty and $M_{Service}^{se}$, M^{se} and M^{te} are empty, it means that the target entity is related to more than one source entity, which fits the Service interface. However, it is necessary to verify if those source entities are related to the target entity only. For that, the process evaluates $M^{se'}$, which corresponds to all matches subtracted from $M_{Service}$ and retrieved from M whose source entities are referred in the matches from $M_{Service}^{te}$. If $M^{se'}$ is empty it means that those source entities relate only to the target entity, thus:

- All matches from k_m and $M_{Service}^{te}$ give raise to a new cluster and are associated with it;

3.3.3 In any other circumstances an n:m relation exists which is not supported by the method. The process proceeds to the next match (step 2).

Step 2 and step 3 are executed until no more matches exist in $M_{Service}$.

As noticed from previous description, only Services with cardinality 1:1, 1:n and n:1 are supported. In fact, on one hand, n:m Services are currently considered unnecessary (5.2.3), and on the other hand Services with cardinality 0:n and n:0 are naturally unsupported by this method since at least one pair of ontologies entities is considered in matches, which collides with the Service interface.

Additionally, notice that only the arguments whose type concerns the ontologies entities are addressed by this method. In fact, the presented method does not try to discover the contents of the arguments whose type is Literal or ArrayOfLiterals since these elements can not be provided by currently envisaged matchers.

The result of this process is a set of clusters, created and validated under the point of view of the interface of the associated Service. Only these clusters and the associated matches are considered for further stages of the automatic bridging process.

7.3.4.1 Example 7.10 – Service-based clustering annotated example

Consider the matches of Table 7.3, previously presented in Table 7.1.

Table 7.3 – Set of matches automatically proposed by matchers

ID	Source entity	Target entity	Matcher	Similarity value	Justifications
m ₁	Individual	Individual	Resnik-like	0.95	∅
m ₂	Individual	Man	Resnik-like	0.86	∅
m ₃	Individual	Woman	Resnik-like	0.86	∅
m ₄	name	given_name	Resnik-like	0.82	∅
m ₅	name	surname	Resnik-like	0.82	∅
m ₆	spouseIn	noMarriages	Resnik-like	0.66	∅
m ₇	name	given_name	MOMIS-like	1	∅
m ₈	name	surname	MOMIS-like	1	∅
m ₉	Individual	Individual	MOMIS-like	0.78	∅
m ₁₀	Individual	Man	MOMIS-like	0.78	∅
m ₁₁	Individual	Woman	MOMIS-like	0.78	∅

Consider also a subset of the Services described in 6.2.3, whose interfaces are briefly described in Table 7.4:

Table 7.4 - Some Services: short description and interface

Argument ID	Type	Semantics
CopyInstance	Creates target concept instances for each source concept instance. This is the service implicitly associated with ConceptBridges.	
Source Concept	Concept	Source ontology concept whose instances will be transformed.
Target Concept	Concept	Target ontology concept to create.
CopyRelation	Creates a relation between target concepts instances.	
Source Path	Path	Source ontology path for each path the bridge will be executed.
Target Path	RelationPath	Target ontology path to create.
CopyAttribute	Copies (no changes) the source property value to the target property instance.	
Source Attribute	AttributePath	Source ontology attribute whose instances will be copied.
Target Attribute	AttributePath	Target ontology attribute to copy to.
CountProperties	Counts the number of instances of a property and creates the target property attribute with that value.	
Source Path	Path	Source ontology path to count.
Target Attribute	AttributePath	Target ontology attribute to create.
Split	Splits by the separators, the source attribute instance into many target attribute instances.	
Source Attribute	AttributePath	Source ontology attribute whose instances will be splitted.
Separators	ArrayOfLiterals	Literals or regular expressions to split by.
Target Attributes	ArrayOfAttributePaths	List of attributes to instantiate with splitted values.

Concatenation	Concatenate several source attribute instances into the target attribute instance, each one separated by a literal.	
Source Attributes	ArrayOfAttributePath	List of source attributes whose instances will be concatenated.
Separators	ArrayOfLiterals	Literals to concatenate between attribute values.
Target Attribute	AttributePaths	Target attributes to instantiate with concatenated values.

The process runs in the scope of every Service of the system, and will generate a set of clusters CL_{O1-O2} . Next sections present the clustering process based on this set of matches and Services. Because the description follows step by step the described process, next sections are numbered according to the steps followed during the execution.

7.3.4.1.1 CopyInstance

1. $M_{CopyInstance} = \{m_1, m_2, m_3, m_9, m_{10}, m_{11}\}$
2. m_1 is the first match to be processed and the respective knot is $k_{m_1} = \{m_1, m_9\}$. $M_{CopyInstance}$ becomes $\{m_2, m_3, m_{10}, m_{11}\}$. $M_{CopyInstance}^{O1:Individual} = \{ \}$, $M_{CopyInstance}^{O2:Individual} = \{ \}$, $M^{O1:Individual} = \{ \}$ and $M^{O2:Individual} = \{ \}$
3. Because the cardinality of CopyInstance is 1:1:
 - 3.1 Because previous sets are all empty, it means that a cluster exists between these two ontologies, and that the CopyInstance Service supports an eventual transformation. In that sense, a new cluster is created ($cl_1 = \langle CopyInstance, \{m_1, m_9\} \rangle$).

The process proceeds to the next match from $M_{CopyInstance}$, which corresponds to execute step 2 of the process. Because processing matches m_2 and m_3 is similar to that of match m_1 , no further description is presented. Yet, the resulting clusters are $cl_2 = \langle CopyInstance, \{m_2, m_{10}\} \rangle$ and $cl_3 = \langle CopyInstance, \{m_3, m_{11}\} \rangle$. Thus, the so far generated clusters are:

$$CL_{O1-O2} = \{cl_1, cl_2, cl_3\}$$

7.3.4.1.2 CopyRelation

1. $M_{CopyRelation} = \{ \}$;
2. Because no match conforms to the type of arguments of the CopyRelation Service, no process will further occur, and therefore no clusters will be generated.

7.3.4.1.3 CopyAttribute

1. $M_{CopyAttribute} = \{m_4, m_5, m_7, m_8\}$

2. m_4 is the first match to be processed in the scope the CopyAttribute Service. Its knot is $k_{m_4} = \{m_4, m_7\}$ thus $M_{CopyAttribute} = \{m_5, m_8\}$. Furthermore, $M_{CopyAttribute}^{O1:name} = \{m_5, m_8\}$, $M_{CopyAttribute}^{O1:name} = \{ \}$, $M_{CopyAttribute}^{O2:given_name} = \{ \}$ and $M_{CopyAttribute}^{O2:given_name} = \{ \}$. Consequently, $M_{CopyAttribute} = \{ \}$;
3. Because the cardinality of the CopyAttribute is 1:1:
 - 3.1 Because $M_{CopyAttribute}^{O1:name}$ is not empty, it means no cluster exists that the CopyAttribute Service is capable to process.

Because $M_{CopyInstance}$ is empty, no further process will occur in the scope of this Service.

7.3.4.1.4 CountProperties

1. $M_{CountProperties} = \{m_6\}$
2. m_6 is the first and unique match in $M_{CountProperties}$ and its knot is $k_{m_6} = \{m_6\}$. Furthermore, $M_{CountProperties}^{O1:spouseIn} = \{ \}$, $M_{CountProperties}^{O1:spouseIn} = \{ \}$, $M_{CountProperties}^{O2:noMarriages} = \{ \}$ and $M_{CountProperties}^{O2:given_name} = \{ \}$. Accordingly, $M_{CountProperties} = \{ \}$;
3. Because the CountProperties Service cardinality is 1:1:
 - 3.1 Because all previous sets are empty, an independent cluster exist that can be processed by the CountProperties Service. A new cluster is then created ($cl_4 = \langle CountProperties, \{m_6\} \rangle$)

Because no more matches exist in $M_{CountProperties}$, no further processing occurs in the scope of this Service. Consequently, the generated clusters correspond now to:

$$CL_{O1-O2} = \{cl_1, cl_2, cl_3, cl_4\}$$

7.3.4.1.5 Split

1. $M_{Split} = \{m_4, m_5, m_7, m_8\}$;
2. m_4 is the first step to be processed thus, $k_{m_4} = \{m_4, m_7\}$ and $M_{Split} = \{m_5, m_8\}$. Furthermore, $M_{Split}^{O1:name} = \{m_5, m_8\}$, $M_{Split}^{O1:name} = \{ \}$, $M_{Split}^{O2:given_name} = \{ \}$ and $M_{Split}^{O2:given_name} = \{ \}$. Consequently, $M_{Split} = \{ \}$;
3. Because the cardinality of the Split Service is 1:n:
 - 3.2 Because $M_{Split}^{O1:name}$ is not empty and the other sets are empty:
 - 3.2.2 Because $M^{te'} = \{ \}$, a new cluster is created ($cl_5 = \langle Split, \{m_4, m_5, m_7, m_8\} \rangle$).

Because no further matches exist in M_{Split} no further process will occur for this Service.

7.3.4.1.6 Concatenation

1. $M_{Concatenation} = \{m_4, m_5, m_7, m_8\}$;
2. m_4 is the first match to be processed thus $k_{m_4} = \{m_4, m_7\}$ and $M_{Concatenation} = \{m_5, m_8\}$.

Additionally, $M_{Concatenation}^{O1:name} = \{m_5, m_8\}$, $M^{O1:name} = \{ \}$, $M_{Concatenation}^{O2:given_name} = \{ \}$ and $M^{O2:given_name} = \{ \}$. As consequence, $M_{Concatenation} = \{ \}$;

3. Because the cardinality of the Concatenation Service is n:1:
 - 3.3 Because $M_{Concatenation}^{O1:name}$ is not empty and the other sets are empty, no cluster exists that can be processed by the Concatenation Service.

Because $M_{Concatenation}$ is empty, no further process occurs in the scope of this Service.

7.3.4.1.7 Example overview

The result of the example is therefore a set of five clusters:

$$CL_{O1-O2} = \{cl_1, cl_2, cl_3, cl_4, cl_5\}$$

Despite the fact the example concerns a very simple case it presents many of the situations arising in more complex scenarios.

7.3.4.2 Service interface conforming vs. non-conforming matches

One of the situations not described in this example concerns the case of entities belonging to some matches that conform to the Service interface, and to others matches that do not conform such interface. The matches between at least one of the entities of m that do not conform to the Service interface are evaluated into M^{se} and M^{te} . According to the described method, the cluster is refused in case M^{se} and M^{te} are not empty. This decision is based on the rationale followed in the rest of the cases that determines that no cluster will be created if no independent set of matches exists. However, another rationale can be followed in this situation, based on the idea that matchers may provide false matches that will distort the process. Following this rationale, such matches should be disregarded.

Because both approaches are justifiable depending on the specific mapping scenario, and because the method is easily configurable, the decision to adopt one or the other approach is up to the user.

The only difference in the description presented above concerns the evaluation of M^{se} and M^{te} , and the evaluation of $M^{se'}$ and $M^{te'}$. In case the second behavior is chosen, M^{se} and M^{te} will not be evaluated nor applied during the process. Furthermore, $M^{se'}$ and $M^{te'}$ will be evaluated without retrieving elements from M . The rest of the process is maintained.

7.3.4.3 CopyInstance specificity

This section addresses some particularities in applying proposed method to the CopyInstance Service.

As described in previous chapters, ConceptBridge and CopyInstance Service are responsible for the creation of target concept instances from either source concept or property. While concept to concept semantic relation poses no problem, the property to concept semantic relation requires some extra attention.

However, when defining a property to concept semantic relation in SBO, the source concept is still mandatory, while the property specification is done through the extensional specification (6.3). This type of semantic relation is not supported by the interface of the CopyInstance Service specified above. As consequence, even in case matches suggest a property to concept semantic relation, no cluster will be generated. Thus, the CopyInstance Services interface must be expanded in order to support this type of semantic relation too.

However, experiences showed that matchers often provide matches between:

- Source concept and target concept;
- Source property and target concept.

Yet, matchers seldom suggest both matches.

In order to support this observation, a disjunctive operator may be stated between the arguments responsible for the inclusion of the source concept and the property into the cluster. However, the disjunctive operator is not provided in the Service specification and therefore it cannot be applied.

Due to the CopyInstance Service specific competencies in the system, it has been decide to provide a special implementation of this method for this Service. Therefore, the Service interface in the scope of the clustering process is defined in three different combinations of arguments (Table 7.6, Table 7.7 and Table 7.8):

Table 7.5 – Initial interface of the CopyInstance Service

Argument ID	Type	Semantics
CopyInstance	Creates target concept instances for each source concept instance.	
Source Concept	Concept	Source ontology concept whose instances will be transformed.
Target Concept	Concept	Target ontology concept to create.

Table 7.6 – CopyInstance Service property to concept interface

Argument ID	Type	Semantics
CopyInstance		Creates target concept instances for each source property instance.
Source Path	Path	Source ontology concept whose instances will be transformed.
Target Concept	Concept	Target ontology concept to create.

Table 7.7 – CopyInstance Service property to concept interface

Argument ID	Type	Semantics
CopyInstance		Creates target concept instances for each source property instance of the source concept.
Source Concept	Concept	Source ontology concept whose instances will be transformed.
Source Path	Path	Source ontology concept whose instances will be transformed.
Target Concept	Concept	Target ontology concept to create.

Because the Service aims to maximize the association of matches, and because the interfaces should be mutually exclusive, the order in which the interfaces are applied is not arbitrary. In particular, the last presented interface (Table 7.7) should be applied first. This interface will be used if matches support the semantic relation between the two pairs of entities. In case this interface does not succeed, it is sure that at maximum one of the others interfaces will succeed. In that sense, the order of application of the other two interfaces (Table 7.5 and Table 7.6) is arbitrary.

Unlike the other Services, whose interface conforms the standard specifications and are therefore automatically supported by the default method (implementation), the CopyInstance Service should provide its own implementation to support the three distinct interfaces.

7.3.4.4 Outlook of the method

The method described in this section aims to reduce the search space of the automatic bridging process by exploiting the interface specification of the available Services. The method is based on four possible types of judgment when processing the matches proposed by matchers. Despite its simplicity, the process can effectively and accurately reduce the set of possible semantic relations (clusters) according to the set of matches.

These clusters will be used in next phases of the automatic bridging process, providing the reasoning elements so the Service can further judge the cluster semantic relevance to the mapping. This is the subject of next section.

Once the relevance is stated, the remaining clusters will give rise to SemanticBridges which will be interrelated to form a valid SBO document. This is the subject of the Section 7.3.6.

7.3.5 Improving Services judgment capabilities

In some circumstances the number of clusters generated for the same set of matches is still very large. These circumstances arise due to the existence of multiple Services with the same interface. When this happens, multiple clusters with exactly the same set of matches will be generated in the scope of different Services. While some SemanticBridges (derived from clusters) may apply the same set of entities, this is not very common and tend to be considered ambiguous. In this sense, new capabilities should be included in Services so they can better judge about the relevance of the SemanticBridge they are proposing.

Semantic bridging is a highly subjective task, demanding extensive domain expertise. Because the goal is to reduce the user participation in the process, user-based domain expertise should be disregarded and substituted by other means. Such expertise is in some extent provided by the matchers and their outcome: the matches. Like any other expertise, it should be adequately applied or it becomes useless.

The goal is therefore to develop a system that combines the Services and Matchers expertise in defining better clusters. Once again, the proposed approach suggests the exploitation of the multi-dimensional service-oriented architecture to this problem, by centering in Services the competencies to better exploit the expertise provided by the matches.

As previously referred, each type of matcher assesses a specific type of similarity between two ontologies entities. Thus, two distinct types of matches between the same pair of entities provide different meanings of the entities relationship and may be applied in distinct circumstances more appropriately than interpreting it equally in all circumstances, as adopted for most of the mapping projects (e.g. [Bergamaschi *et al.*, 1999; Doan *et al.*, 2002; Madhavan *et al.*, 2001; Miller *et al.*, 2000; Mitra *et al.*, 1999]).

Accordingly, it is our conviction that adopting correctly and precisely the meaning of the matches to each circumstance would provide good disambiguation results, reducing even more the combinatorial space in order to propose relevant SemanticBridges.

7.3.5.1 Proposed approach

Services must therefore determine the conditions in which the entities seem to form a relevant SemanticBridge. In particular, Services are requested to specify the following parameters:

- The types of matches exploited by the Service;
- The maximum and minimum admissible values (thresholds) for the match values. Thresholds can be defined in respect to each specific type of match or generically;
- The elements that should be provided through the justification element of matches;

- The mathematical expression to combine the matches values, into a judgment value;
- A decision expression capable to decide about the accuracy of the cluster.

Example 7.11 – Improving Services judgment capabilities

Table 7.8 presents the simple parameterization of seven Services. In this parameterization, the three matchers presented in 7.3.3.1 are applied.

Table 7.8 – Automatic semantic bridging Services requirements

Service	Considered Matchers	Threshold	Justifications	Combination and Decision expression
CopyInstance	Resnik-like	$0.7 < X \leq 1$		average of matches values > 0.8
	MOMIS-like	$0.7 < X \leq 1$		
CopyRelations	Resnik-like	$0.5 < X \leq 1$		average of matches values > 0.7
	MOMIS-like	$0.7 < X \leq 1$		
CopyAttribute	Resnik-like	$0.8 < X \leq 1$		average of matches values > 0.85
	MOMIS-like	$0.8 < X \leq 1$		
Split	Resnik-like	$0.5 < X \leq 1$		average of matches values > 0.7
	MOMIS-like	$0.7 < X \leq 1$		
	Type-checker	$1 \leq X \leq 1$	[type=="string"]	
Concatenation	Resnik-like	$0.5 < X \leq 1$		average of matches values > 0.75
	MOMIS-like	$0.7 < X \leq 1$		
	Type-checker	$1 \leq X \leq 1$	[type=="string"]	
CountProperties	Resnik-like	$0.6 < X \leq 1$		average of matches values > 0.7
	MOMIS-like	$0.8 < X \leq 1$		
	Type-checker	$1 \leq X \leq 1$	[type=="property-number"]	
Currency Converter	Resnik-like	$0.3 \leq X \leq 0.5$		average of matches values > 0.3
	Type-checker	$1 \leq X \leq 1$	[type=="currency"]	

The simplest parameterization of Services states that for every pair of ontologies entities, one match of every defined type conforming to the threshold values should exist.

Example 7.12 – Confirming a cluster according to the Service requirements

Consider the following cluster, already suggested in 0:

$$cl_1 = \langle CopyInstance, \{m_1, m_9\} \rangle$$

According to previous table, every pair of entity adopted in the cluster with the CopyInstance Service, should be supported by two distinct matches:

- Resnik-like matches, whose similarity value (X) should conform to the expression $0.7 < X \leq 1$;
- MOMIS-like matches, whose similarity value (X) should conform to the expression $0.7 < X \leq 1$.

Because the cl_1 cluster relates O1:Individual to O2:Individual entities, it means that one match of each of previous types should exist. Cluster cl_1 associates two matches, which are the only matches to consider now. These matches are represented again in Table 7.9.

Table 7.9 – O1:Individual-O2:Individual matches

ID	Source entity	Target entity	Matcher	Similarity value	Justifications
m ₁	Individual	Individual	Resnik-like	0.95	[]
m ₉	Individual	Individual	MOMIS-like	0.78	[]

In fact, the matches in cluster cl_1 fulfill the Service requirements concerning the type and value. The mathematical expression is evaluated, resulting in the value 0.865. This value is further applied in the decision expression, which determines that the cluster is relevant ($0.865 > 0.8$) and is therefore maintained in the clusters list.

Example 7.13 – Dismissing a cluster according to the Service requirements

Unlike previous cluster, when processing cluster $cl_4 = \langle \text{CountProperties}, \{m_6\} \rangle$, the CountProperties requirements are not fulfilled. In particular, no match of Type-checker type exists in the cluster, but is required by the Service. In that sense, the cluster is discarded from the list.

Instead of obliging that every pair of entities is supported by all specified matches, the Service parameterization can be improved by defining less restrictive conditions in the form of logical operations.

Example 7.14 – Refining Service requirements

The CountProperties Service can be re-parameterized such the MOMIS-like and Resnik-like matches are disjunctive with respect to the Type-checker matches:

$$\text{or}(\text{Type-checker}, \text{and}(\text{MOMIS-like}, \text{Resnik-like}))$$

Nevertheless, even with this parameterization, the process described in Example 7.13 would fail. In fact, two matches (MOMIS and Resnik-like) are necessary while the cluster provides only one (m_6) of the Resnik-like type.

7.3.5.2 Outlook of the proposed approach

Despite the reduced number of performed experiences, the developed method indeed reduces ambiguity of the proposed clusters, motivating further experiences in order to clearly determine potentialities and limitations. However, it is already noticeable the limitations arising from the subjective nature of the matches. In fact, matches are calculated according to and based in statistical and often subjective information. In particular, WordNet-based matchers (e.g. [Resnik, 1999]) often exploit subjectively annotated *corpus* and apply statistic-oriented inferences, leading to intrinsically subjective matches. In that respect, matchers based in formal information should be applied (and eventually developed) into the system, providing more deterministic matches, leading therefore to more accurate results.

7.3.6 Automatic definition of the ontology mapping document

Despite the possibility to improve decision capabilities of Services, at some point it is necessary to transform clusters into a valid and meaningful ontology mapping document.

The methodology presented in this section aims to transform the clusters remaining from the clustering phases into a valid ontology mapping document defined according to the Semantic Bridging Ontology (SBO). Therefore, the automatic bridging process should follow an object-oriented, property-centric methodology, as promoted by ontologies and SBO. Yet, the process is not unique and therefore some heuristics are applied during the process.

The process concerns with the definition of five distinct elements:

1. Definition of ConceptBridges;
2. Definition of relationships between ConceptBridges (\prec -relation);
3. Definition of PropertyBridges;
4. Definition of relationships between ConceptBridges and PropertyBridges (\diamond -relation);
5. Definition of AlternativeBridges and their relations with ConceptBridges and PropertyBridges.

The process runs in this order and is therefore task oriented instead of cluster-oriented, as would suggest the presence of the clusters.

7.3.6.1 Definition of ConceptBridges

The process defines ConceptBridges according to the clusters with the CopyInstance Service associated. For every of such clusters, one of three distinct processes will be executed, depending on the interface of the Service (refer to 7.3.4.3).

1. In case the cluster relates two concepts, the process is rather simple. A ConceptBridge should be created between the source and target concepts;
2. In case the cluster relates a property to a target concept, it is necessary to find (decide) which source concept is semantically related to the target concept. Currently, the decision is taken depending on two situations:
 - 2.1 If at least one of the domain concepts of the property is already semantically bridged, one is chosen arbitrarily;
 - 2.2 If none of the property domain concepts is semantically bridged, one of its domain concepts is arbitrarily chosen.

In future research and development of the method, other heuristic rules may be applied. For example, it is envisaged the exploitation of the MOMIS (matcher) information about the analysis of the relations between ontologies entities, which might provide useful hints;

3. In case the cluster relates a concept and property to a target concept, the process creates a ConceptBridge that, besides the definition of the source and target concepts, defines an

extensional specification element based on the source property. The definition of the extensional specification elements poses some difficulties due to the fact that matches refer to properties independently of their domain and range (e.g. O1:name instead of O1:Individual/name/Literal). However, because properties are always applied in SemanticBridges through Paths, it is necessary to calculate the Path such:

- The last property of the Path is the source property specified in the cluster;
- The root concept of the Path is the source concept specified in the cluster.

Because in “typical” ontologies properties do not have a large number of domain concepts, the Path definition is performed exhaustively with fair performance. However, this process applies some heuristics and is therefore subject of improvements in future stages of research.

7.3.6.2 Definition of \prec -relationships

Once ConceptBridges are completely defined according to previous rules, it is time to define the \prec -relationships. The definition of \prec -relationships follows the rule/constraint about this relation specified in 5.4.7.4:

$$\begin{aligned} & \forall cb_1, cb_2 \in \mathcal{B}^c \\ & \quad \prec(cb_2, cb_1) \\ \Rightarrow & \quad sConcept(cb_1, c_1^s) \wedge tConcept(cb_1, c_1^t) \wedge sConcept(cb_2, c_2^s) \wedge tConcept(cb_2, c_2^t) \wedge \\ & \quad \left((is_a(c_2^s, c_1^s) \wedge is_a(c_2^t, c_1^t)) \vee (is_a(c_2^s, c_1^s) \wedge c_1^t = c_2^t) \vee (c_1^s = c_2^s) \wedge is_a(c_2^t, c_1^t) \right) \end{aligned}$$

If the evaluation of the right part of the rule to two ConceptBridges holds true, the left side conclusion is drawn and the \prec -relationships is established between the two ConceptBridges.

The relationship defined in this phase is exploited in next phases of the process, especially to better:

- Calculate the Paths in PropertyBridges;
- Determine the \diamond -relationships (between ConceptBridges and PropertyBridges).

7.3.6.3 Definition of PropertyBridges

The main problem in defining PropertyBridges from clusters concerns the calculation of the Paths according to the properties defined in clusters. The adopted process is similar to that described for the extensional specification element of the ConceptBridges. However, the process is constrained by other facts, namely the existence of \prec -relationships and the need to define \diamond -relationships. In this sense, the definition of PropertyBridges and the definition of \diamond -relationships cannot be dissociated from each other.

The adopted approach is based on the fact that the target properties should be applied through a one-step Path (5.4.7.3). In result of this constraint, it is mandatory that the domain concept of the target properties is semantically related to one source concept. In case target properties do not have the same domain concept the cluster is discarded since it is not supported as it is suggested by the cluster. Otherwise, two situations may occur:

1. If the target concept is already semantically related (to a source concept c_1), then the source Paths should all have c_1 as root concept. In case multiple SemanticBridges exists to target concept, it is chosen the one that provides the shortest Paths to the properties;
2. If the target concept is not yet semantically related, then a ConceptBridge is inferred between the target concept and the domain concept of the source properties. Besides the generation of the inferred ConceptBridge, in order to maintain the automatic bridging system coherent, the following elements are also generated and inserted in their respective repositories:

- $match_i = \langle source\ concept, target\ concept, inferred, 1, [] \rangle$
- $cl_i = \langle CopyInstance, \{match_i\} \rangle$

Example 7.15 – Inferring matches between concepts based on PropertyBridges

Consider the abstract ontology mapping scenario presented in Figure 7.4. Each one of the two represented attributes of source ontology (SO namespace) matches the same attribute of the target ontology (TO namespace). Yet, matchers did not provide matches between concepts or between property and concept.

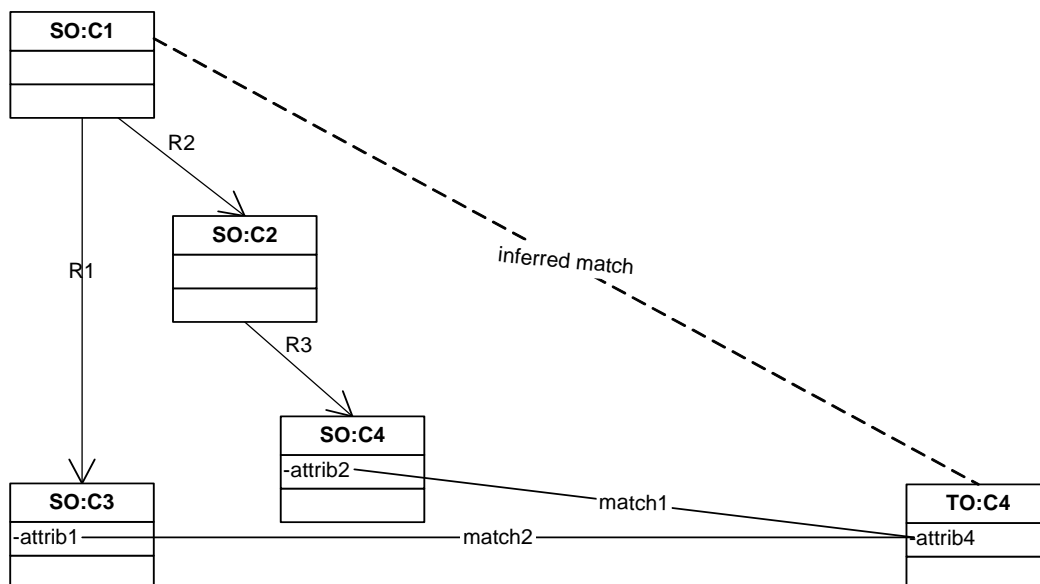


Figure 7.4 – Abstract scenario representing an inferred match

In this particular case there is only one concept providing a common root for the Paths: SO:C1. Accordingly, a new match between SO:C1 and TO:C5 is inferred, and a new cluster is generated. Thus, the calculated Paths are:

$$W_{attrib1}^s = SO : C1 / R1 / C3 / attrib1 / Literal$$

$$W_{attrib2}^s = SO : C1 / R2 / C2 / R3 / C4 / attrib2 / Literal$$

$$W_{attrib4}^t = TO : C5 / attrib4 / Literal$$

Despite the uncommon situation, it may happen that not all source properties defined in the cluster have the same domain concept. In such circumstances it is necessary to search for a concept from which it is possible to reach all the source properties (i.e. the common root concept for all Paths). In case multiple root concepts exist, it is selected the one that provides the shortest Paths to the properties.

Notice that due to this constraint, a common concept may not exist. In such cases the proposed clusters are not transformed into SemanticBridges and are instead discarded⁴⁶.

7.3.6.4 Definition of \diamond -relationships

Once the PropertyBridges are specified, the specification of \diamond -relationships is quite necessary. Basically it consists in \diamond -relate every PropertyBridge (pb_1) with the ConceptBridge (cb_1) whose concepts are the root concepts (source and target concept) of the Paths (source and target Paths respectively) of pb_1 .

Yet, a complementary task is performed at this stage. It concerns the use and promotion of the features provided by the \prec -relation. Basically, it concerns the modification of the PropertyBridges Paths so the root concepts of the Paths are substituted by their super-concepts. This modification is performed according to the constraint presented in 5.4.7.4 and re-presented in 7.3.6.2.

Thus, if none of the source or target Paths is modified previous solution prevails. However, even if only one of the Paths is modified, some actions are performed:

- The modified Paths are applied;
- The PropertyBridge is re- \diamond -related with the ConceptBridge cb_2 , such $\prec(cb_1, cb_2)$.

It might occur though, that cb_2 does not exist so far. In such case, the ConceptBridge is created (inferred) and the corresponding match and cluster are derived from it and pushed into the respective repositories.

Example 7.16 – Definition of \diamond -relationships

Consider the matches presented in Table 7.10 respecting the scenario of Figure 7.2. This set of matches is a subset of those presented in Table 7.1 in order to demonstrate previous process. The proposed matches ignores the similarities between O1:Individual and O2:Individual and between O1:spouseIn and O2:noMarriages.

⁴⁶ In current implementation of the method, no backward Paths are calculated, but they may be useful in cases no common root concept is found in forward Paths.

Table 7.10 – Limited set of matches for the ontology mapping scenario of Figure 7.2

ID	Source entity	Target entity	Matcher	Similarity value	Justifications
m ₂	Individual	Man	Resnik-like	0.86	∅
m ₃	Individual	Woman	Resnik-like	0.86	∅
m ₄	name	given_name	Resnik-like	0.82	∅
m ₅	name	surname	Resnik-like	0.82	∅
m ₇	name	given_name	MOMIS-like	0.81	∅
m ₈	name	surname	MOMIS-like	0.85	∅
m ₁₀	Individual	Man	MOMIS-like	0.78	∅
m ₁₁	Individual	Woman	MOMIS-like	0.78	∅

In case the last proposed process is not executed, the proposed matches would result in the ontology mapping document represented in Figure 7.5⁴⁷.

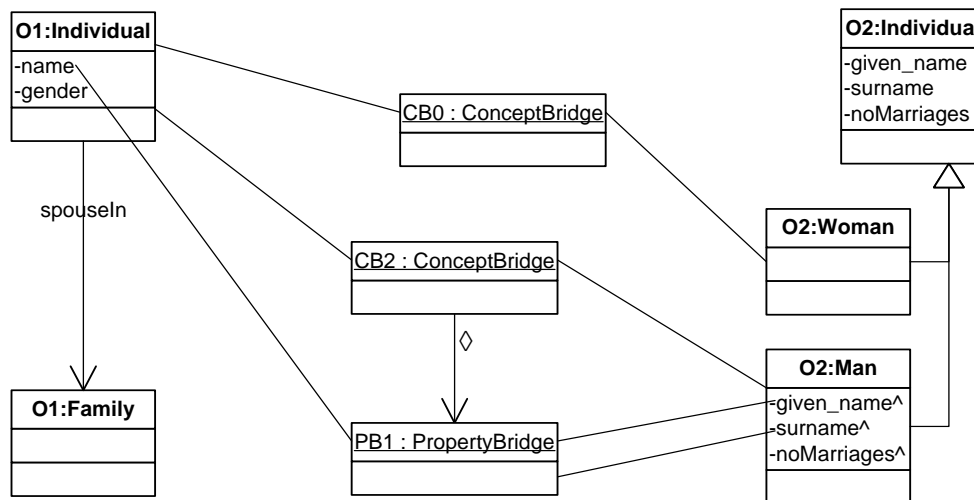


Figure 7.5 – Automatic semantic bridging without exploiting the properties inheritance

Instead, if the proposed process is performed, the resulting ontology mapping document would be that represented in Figure 7.6.

Notice that in first solution only O2:Man instances will be filled in with the given_name and surname attribute values. In the second solution instead, because the PropertyBridge is \diamond -related with the super ConceptBridge, both O2:Woman and O2:Man will profit from the PropertyBridge.

⁴⁷ Notice that the ^ symbol has been appended to the inherited properties in order to stress the inherited relation. While not correct UML notation, it is considered beneficial to the comprehension of the example)

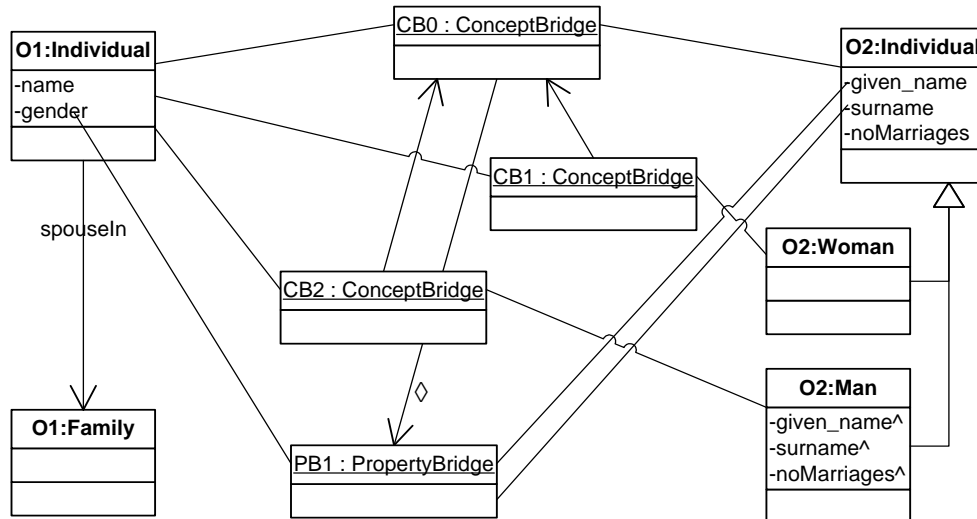


Figure 7.6 – Automatic semantic bridging when exploiting the properties inheritance

7.3.6.5 Definition of AlternativeBridges

The definition of AlternativeBridges is limited to AlternativeBridges-of-PropertyBridges. Their definition is not mandatory but it promotes expressiveness and clarity of the proposed ontology mapping document.

In particular, PropertyBridges are defined alternatives in the scope of an AlternativeBridge-of-PropertyBridges (\perp^{B^P}) in case the properties (Paths) of two or more PropertyBridges are exactly the same. In such circumstances it is assumed that the Services that suggested such PropertyBridges are very similar and are commonly applied one instead of the other.

Example 7.17 – Services commonly used alternatively

Both the Split and the Split-By-Regular-Expression Services divide an attribute instances into multiple fragments. However, while the first divides the string by constant literals, the second makes use of a regular expression.

This decision is based on the evidences arising from experiences performed in various ontology mapping scenarios. Still, it is just a heuristic rule that might be modified or improved.

7.3.6.6 Outlook of the automatic definition of the ontology mapping document

The automatic definition of the ontology mapping document is based on the automatically proposed clusters, which in turn are calculated mostly according to automatically generated matches.

A set of competencies gained from the manual, user-based experiences have been systematized into a whole coherent process. Yet, these competencies are fundamentally heuristic-based which has considerable benefits when ontology mapping scenarios do not diverge significantly from those done manually.

7.3.7 Outlook of the automatic bridging process

This section described the automatic bridging system researched and developed as a case test of the proposed and described Multi-Dimensional Service-oriented Architecture.

In the scope of this thesis, matches are understood as rudimentary expert opinions about the semantic relation between a source ontology entity and a target ontology entity. In the lack of better opinions, instead of adopting a single type of match or a set of type of matches, the proposed approach promotes the application of multiple distinct types of matches, which can be included into and exploited by the system as required.

Because distinct types of matches provide different expertise opinions, in order to capture maximum benefices from each type, Services define their own constraint requirements according to matches types, values and their combination. Due to the Services features, respecting competencies and independence preconized through the multi-dimensional service-oriented architecture, Services are able to customize their requirements independently of the others and in a more versatile manner.

The process itself comprehends three distinct phases:

- The Service-interface clustering phase, provides a fast and reliable association of matches with Services (clusters);
- The Service-constraints re-clustering phase, exploits the Service-defined matches constraints to judge upon the semantic relevance of the clusters suggested from previous phase. The result is a set of fully qualified clusters;
- The cluster-based automatic definition of the ontology mapping document phase transforms clusters into SemanticBridges and create relationships between them. During the Path calculation and the relationships specification, new semantic relations are inferred and transformed into ontology mapping document elements. For backward compatibility, matches and clusters are generated from these inferred elements, promoting feedback and coherency between ontology mapping process phases.

In all the phases though, the proposed approach is strongly influenced by heuristic rules gained and systematized from the manual bridging experiences performed during this research. The ontology mapping document is therefore the result of a set of judgments and decisions, which like any others are fallible and arguable. Yet, even if at small scale, any automation or driving support of the semantic bridging process is of great help for the domain expert. Moreover, the domain-expert has the ultimate decision about the contents and semantic correctness of the ontology mapping document.

7.4 Summary

The Multi-dimensional Service-oriented Architecture advocates that ontology mapping system capabilities and its supported semantic relations are ultimately dependent on the type of transformations allowed/available in the system. Services represent the transformation capabilities in SBO, in semantic bridging and in the execution system, but the proposed architecture suggests that their capabilities should be expanded to support the requirements of other phases of the process.

Accordingly, Services embody useful and eventually fundamental competencies for distinct phases of the process, that were originally an exclusive competence of the domain expert. The domain expert know-how is therefore acquired and integrated into the system. Yet, instead of a monolithic structure representing such knowledge, multiple independent and dynamically evolving modules are used. These modules however, instead of adopting a task-oriented structure, are orthogonal to multiple phases of the ontology mapping process providing different functionalities depending on the requesting phase. This coincides with the MAFRA ideas presented in Chapter 4. In fact, Services represent some of the entities composing the Domain Knowledge & Constraints module, which are potentially inter-related with all core phases of the ontology mapping process (Figure 7.7).

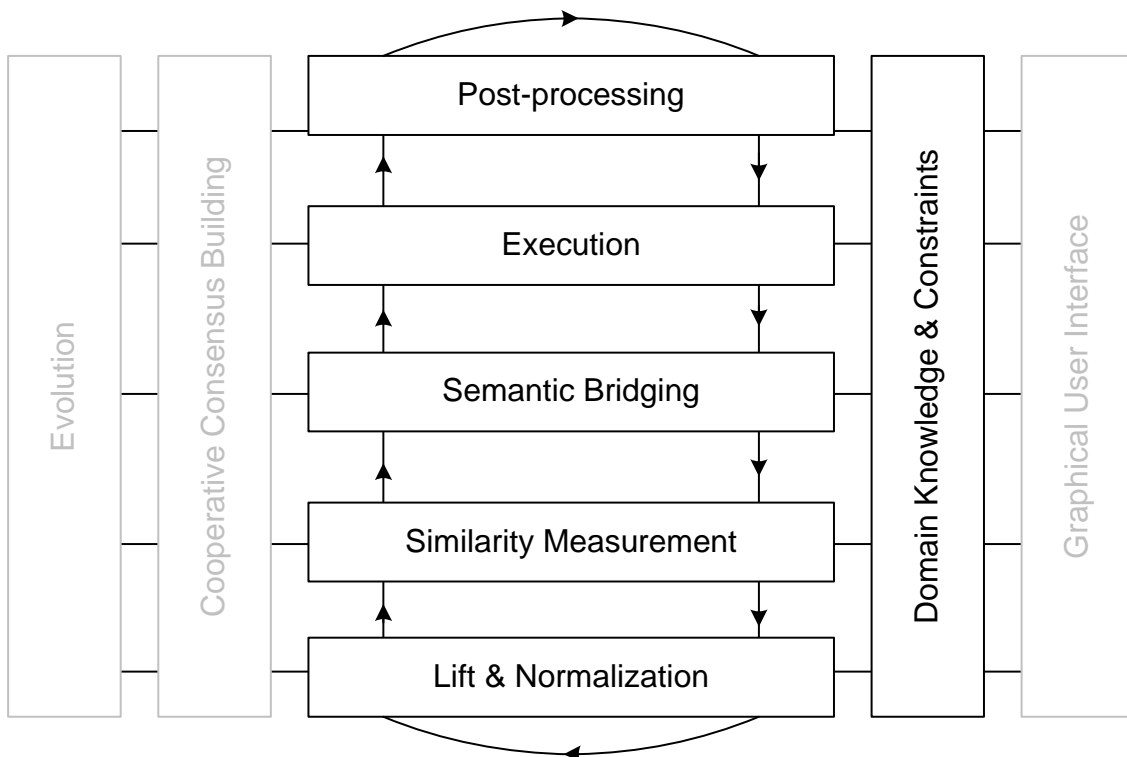


Figure 7.7 – MAFRA emphasis on Domain Knowledge & Constraints module relations

While the proposed architecture is generically applicable in all ontology mapping process phases, the preconized ideas have been adopted and exploited in the automation of the semantic bridging phase as a case test.

The proposed automatic bridging process suggest that every Service defines the conditions in which certain set of source and target ontology entities have good probability to be semantically related through the Service. Combining the information resulting from general, wide purpose matchers with a set of heuristic-based rules, the system is able to propose a coherent and valid ontology mapping document.

The main lack in the research and development of the automatic semantic bridging process is the lack of formal and reported experiences. This is especially due to the three following facts:

- Inexistence of a battery of ontology mapping tests commonly used by the research community;
- Inexistence of similar reports by other research teams, which in turn is due to the;
- Inexistence of similar systems or approaches.

Yet, the system implemented according to the proposed process performs well and provides the user with a good starting ontology mapping document for further improvements.

Chapter 8

DEVELOPMENT AND EXPERIENCES

This chapter describes pragmatic issues concerning the development of the proposed research ideas into a usable and useful ontology mapping system tool. The work described in this chapter has been previously described, namely in [Silva *et al.*, 2003; Silva & Rocha, 2003c; Silva & Rocha, 2003e; Silva & Rocha, 2004a; Silva & Rocha, 2004b].

The resulting tool is being applied in a variety of third party research projects, which provide valuable feedback on the relevance and usability of the research ideas presented in previous chapters. Later on, a simple evaluation and comparison of performance is described.

8.1 Development

In order to test and validate the research proposals of previous chapters, it has been decided to develop a tool that implements such ideas. The implemented tool, named MAFRA Toolkit, is one of the major outcomes of this thesis and is publicly available at [MAFRA Toolkit].

Due to the multiple heterogeneous phases encompassed in the ontology mapping process, multiple technologies are required and have been used. In order to systematize the implementation process, four subjects are fundamental:

- Ontology and Knowledge-base manipulation, that describes the analysis and decision process concerning the adoption of the technology to manipulate ontologies and knowledge bases;
- The semantic bridging implementation, that concerns with the definition and validation of the ontology mapping document according to the Semantic Bridging Ontology constraints;
- The execution process implementation, concerning with the execution process and the transformation Services;
- The Graphical User Interface implementation, which describes not only the interface but the different stages it passed through.

Next sections address each of these subjects.

8.1.1 Ontology and Knowledge Base manipulation

One of the most important decisions respecting the implementation concerns with the adoption of the language for representation and further manipulation of ontologies and knowledge bases. Multiple languages are nowadays available, which motivated a careful analysis of features/requirements and technological support. The analysis, comparison and conclusions are available in the SANSKI Project Report 1.1 [Silva, 2002a]. Pragmatically, several statements have been drawn:

- No standard representation language existed in the early stages of the thesis. RDFS representation framework is the basic representation mechanisms for the Semantic Web to which all Semantic Web ontology representation languages should ground (Figure 8.1).

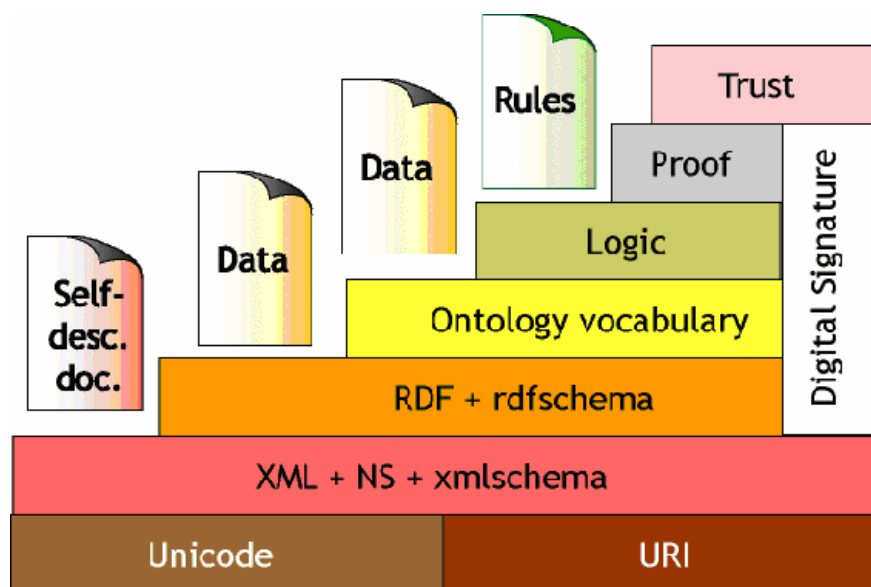


Figure 8.1 – Semantic Web technological layers according to Berners-Lee

Currently, the World Wide Web Consortium⁴⁸ (W3C) recommended OWL (Ontology Web Language) for the ontology representation language in the Semantic Web. OWL is very similar to the DAML+OIL representation language, which grounds on the RDFS representation framework and in Description Logics (DL) theory [Silva, 2003];

- Lack of tools for the manipulation of ontologies and knowledge bases. Most of the tools (e.g. parsers and inference engines) support RDFS but not any further improvements concerning with the ontology model;
- Description Logics has been adopted in most of the ontology representation languages for the Semantic Web (e.g. OIL, DAML, DAML+OIL and OWL). DL is associated with great demands for computational power which is unfeasible in current stage of Semantic Web. OWL Lite has been suggested to overcome such constraints by discarding the Description Logic features. However, OWL Lite is less more than RDFS, to which some cardinality constraints and semantics have been added;
- Ontologies currently available in the Web are mostly developed and specified by exploiting RDFS features only. In fact, nowadays ontologies are typically very simple, based on hierarchy of classes, their attributes and inter-relations between classes. A few of them constrain the cardinality of properties but even this characteristic is poorly exploited.

Accordingly, it became clear that RDFS should be considered the minimum common element of the ontology and knowledge base representation languages.

First development efforts should focus in choosing the technological support, such that it:

- Supports RDFS representation language;
- Abstract RDFS or/and other representation languages into a common and generic manipulation interface;
- Is based in widely spread, open-source code so it is possible to change it according to requirements, namely respecting the ontology and knowledge based formalization presented in Chapter 3;
- Is aware of the ontology and knowledge base representation languages and eventually other Semantic Web technological standards.

Several open source Java-based solutions exist respecting previous requirements (e.g. JENA⁴⁹, ICS-FORTH RDFSuite⁵⁰, or the RDF API from Sergey Melnik⁵¹). Yet, those solutions correspond

⁴⁸ The World Wide Web Consortium is the international organisation responsible for the recommendation and promotion of technology for the Word Wide Web in which the Semantic Web is included.

⁴⁹ <http://jena.sourceforge.net/>

⁵⁰ <http://139.91.183.30:9090/RDF/>

more to a library of competencies to manipulate RDFS documents than ontologies. In first stages of development, adaptation of JENA package has been envisaged, but in meantime it has been realized that a more ontology oriented solution is preferable.

The KAON Workbench [KAON] had risen as the most complete solution found at that time, encompassing not only the ontology and KB manipulation requirements described above, but other very interesting features:

- Stable and efficient ontology manipulation API, wrapping RDFS or any other ontology representation language supported;
- Well established semantics of ontology representation elements [Motik *et al.*, 2003];
- Multi-lingual support layer on top of RDFS;
- OWL Lite like cardinality capabilities;
- Inverse, transitive and cardinality constraints [Motik *et al.*, 2003];
- Ontology development tools, including a graph-based user interface;
- Generic graphic-based entities manipulation library;
- Support for very large ontologies and knowledge bases due to the well established relational data base technology adopted;
- Database to ontology conversion tool;
- Tools for ontology acquisition from text.

KAON is therefore a very powerful solution for ontology manipulation, both in research experience-oriented projects and in the scope of very demanding enterprise driven projects. What is more, KAON is being developed by a very motivated competent research team with direct application in (and feedback from) commercial applications. Deeper details about KAON can be found in [Motik *et al.*, 2003], while the KAON framework is publicly available at [KAON].

8.1.2 Semantic bridging

Once the representation language and supporting tools were selected, the semantic bridging phase implementation corresponds to two distinct tasks:

- The user interface, which is responsible for the definition of the ontology mapping document, described in section 8.1.4;
- Verification of the mapping document, during both the loading and specification processes.

Verification of the mapping corresponds to check the constraints holding between SBO entities as been specified in 5.4. Because KAON is a Java-based system without support for constraint-based

⁵¹ <http://www-db.stanford.edu/~melnik/rdf/api.html>

programming, previously defined constraints have been explicitly and procedurally coded into a hierarchy of Java classes, wrapping not only the SBO concept and its properties, but also their constraints.

Basically, every SBO concept is an implementation of the SBOEntity interface, which define, between others, the method “public void verification() throws Exception” responsible for the verification of the entity contents, relationships and respective constraints.

When loading an existent mapping file, the validation process is triggered by the mapping object (instance of the Mapping class, corresponding to the M SBO concept) as soon as the file is loaded. The verification method is called for every object in the mapping (e.g. ConceptBridges, PropertyBridges, Services), followed by the verification of relationships.

The verification process during the semantic bridging specification is performed in three situations:

- Creation of a new instance of an SBO entity;
- Definition of an attribute for an SBO entity instance;
- Definition of a relationship between two instances of two SBO entities.

The user interface procedure, supporting the change, is responsible for calling the verification method of the instance experiencing or triggering the changes. The verification process, as previously described, is performed in the scope of such instance.

In third situation, changed instance is additionally responsible for calling the verification method of the other instance in the relationship, so it can also verify its changes.

Example 8.1 – Verification process when \diamond -relating two SemanticBridges

When \diamond -relating a PropertyBridge to a ConceptBridge, the PropertyBridge instance is considered the one triggering the changes. Thus, its verification method is called by the user interface responsible method, but the verification method of the PropertyBridge instance is also responsible for calling the verification method of the ConceptBridge instance.

A simple flag mechanism is used to prevent circular callings of verification procedures between entities.

8.1.3 Execution engine

The execution engine is the core element on the MAFRA Toolkit. In fact, while the semantic bridging process can eventually be performed using a simple text editor and user-based verification, the automation of the execution process is the ultimate goal of the ontology mapping process.

Mapping document encompasses all the necessary information to transform instances of source knowledge base into target knowledge base instances. However, unlike other approaches that use generic rules and reasoning engines, SBO is not defined in any rule based language nor is directly

understood by any generic reasoning engine. In that sense, a specific execution engine is necessary to interpret and run the described semantic relations.

As referred in 6.2 the execution process comprehends three phases: query, filtering and transformation and instantiation, which have distinct requirements and demand distinct efforts.

8.1.3.1 Tree-based query

One of the most important limitations of KAON is the inexistence of an ontology query language, which is of fundamental importance in the implementation of the query and filtering phases. In fact, as presented in 6.2.1, the developed process makes extensive use of query language functionalities.

Unfortunately, no filtering, selection, projection, Cartesian product or Join operations are directly supported by KAON. Querying capabilities of KAON are rather limited, allowing only simple questions upon concept and properties instances. Basically, KAON query capabilities can be synthesized into:

- Query a concept instance for all its properties instances, which results into a table-based representation of the concept instance;
- Query the concept instance for the instances of a specific property, which corresponds to a specific column of previous table.

Another very important feature missing in KAON is the multi-step Path support. In fact, because the Path concept is not a basic element in RDF(S), it was not expected to be supported in KAON. Because a Path instance reflects a specific view of the knowledge base, it is directly dependent on the query language, which turns to be considerable difficult to implement due KAON limited query capabilities.

Fortunately, the “backward” query of single Step Paths is directly supported by KAON. Thus, it has been necessary only to adapt it to multi-Step backward Paths.

Therefore, most of the required query operations, including the primitive relational operations, had to be implemented in the scope of this thesis. These operations provide the basic constructs for the implementation of the tree-based representation of Paths and respective knowledge base query, which correspond to one of the most important contributions of this thesis and of MAFRA Toolkit.

8.1.3.2 Filtering

As referred, the filtering operation is not directly support by KAON which compel to its implementation during this thesis.

The filtering process runs for every row of the query table, and concerns with the application of the ConditionExpressions. This corresponds to:

- Instantiate the ConditionExpression with the table values corresponding to the Paths in the ConditionExpression;
- Evaluate every comparison, which results in an instantiated Boolean expression;
- Draw a conclusion (Boolean value) from the instantiated Boolean expression.

The operation is rather simple, but depends both on the comparison and Boolean operators. While Boolean operators are predefined and fixed, comparison operators might vary substantially. A few comparison operators have been defined since the early phases of the project but more comparison operators could be necessary or advisable during the life-cycle of the system.

In that sense, comparison operators should be implemented using the same modular, open approach suggested for SBO and MAFRA, but specially used in Services. Operators are therefore implemented as distinct classes of the engine, characterized and made available to the system through a simple description mechanism, much like Services (7.2).

Example 8.2 – RDF definitions of the Equal and Less Operators

The following RDF code describes two instances of the Operator concept: the Equal and Less:

```
<SBO:Operator rdf:ID=="=">
<rdfs:label>==</rdfs:label>
<rdfs:comment="are two operands equal"/>
<SBO:location="pt.ipp.isep.gecad.mafra.engine.operators.Equal"/>
</SBO:Operator>

<SBO:Operator rdf:ID="Less">
<rdfs:label><</rdfs:label>
<rdfs:comment="is first operand less than second operand"/>
<SBO:location="pt.ipp.isep.gecad.mafra.engine.operators.Less"/>
</SBO:Operator>
```

The name of the Operator is denoted by the value of the rdfs:label property. This corresponds to a string that will be used in the user interface. The Equal operator is denoted by the “==” string, and the Less operator is denoted by “<<”.

A description and competency of the Operator is optionally associated through the rdfs:comment property.

Finally the class implementing the comparison is specified through the SBO:location property, providing the necessary information for the comparison engine to locate and access it.

Both query and filtering processes are implemented in the context of transformation Services, as described in Chapter 6.

8.1.3.3 Transformation engine

The transformation engine is responsible for the core logic of the transformation process. It is implemented in the Engine class of the `pt.ipp.isep.gecad.mafra.engine` package. It is a rather simple process, described by the following Java code:

```
public transformation(Mapping m_mapping)
{
    // read source and target ontologies
    readOntologies(m_mapping);

    // checks mapping correctness
    m_mapping.verification();

    // load source knowledge base instances
    readKnowledgeBases(m_mapping);

    // open target knowledge base for writing
    openKnowledgeBases(m_mapping);

    // initializes TI^2 table
    TransformationInformationTable TI2 = initializesTI2(m_mapping);

    // open the log file for writing
    openLogFile(m_mapping);

    // creates target concept instances
    runConceptBridges(m_mapping, TI2);

    // create target properties values
    runPropertyBridges(TI2);

    // save target instances in target KB
    saveTargetInstances(TI2);
}
```

Despite the self-describing code just presented, the `runConceptBridges` and `runPropertyBridges` procedures deserve a closer description.

The `runConceptBridges` procedure corresponds to the following Java code:

```
private void runConceptBridges(
    Mapping m_mapping,
    TransformationInformationTable TI2) throws Exception
{
    Set setConcepts = m_mapping.getAllSourceConcepts();
    Iterator it = setConcepts.iterator();
    for(; it.hasNext(); ) {
        Concept concept = (Concept) it.next();
        createTargetInstancesOf( concept, TI2 );
    }
}

private void createTargetInstancesOf(
    Concept concept,
    TransformationInformationTable TI2) throws Exception
{
    Set setInstances = concept.getInstances();
    Iterator it = setInstances.iterator();
    for(; it.hasNext(); ) {
        Instance instance = (Instance) it.next();
        Set setCBs = m_mapping.getCBsForSourceConcept( concept );

        Iterator itCBs = setCBs.iterator();
        for(; itCBs.hasNext(); ) {
            ConceptBridge CB = (ConceptBridge) itCBs.next();
            CB.createTargetInstance( instance, TI2 );
        }
    }
}
```

Every concept instance of the source knowledge base is dispatched to every ConceptBridge that semantically relates the concept of the instance. Every ConceptBridge will call the CopyInstance Service, which will be responsible for the query, filtering and transformation phases according to the argument values of the ConceptBridge. This includes the association of extensional specification information in TI^2 .

The runPropertyBridges procedure is responsible for the execution of all PropertyBridges \diamond -related with the ConceptBridge that created the target concept instance, as stored in the TI^2 . The ConceptBridge is responsible for determine such PropertyBridges, which includes not only those

that are directly \diamond -related with the ConceptBridge but also those that are \diamond -related with its super ConceptBridges. The following code represents this process:

```
private void runPropertyBridges (
    TransformationInformationTable TI2) throws Exception
{
    Iterator it = TI2.iterator();

    // for every target instance created previously
    for(; it.hasNext(); ) {
        TransformationInformation TI =
            (TransformationInformation) it.next();
        ConceptBridge CB =
            m_mapping.getConceptBridge( TI.getConceptBridge() );
        CB.runPropertyBridges( TI );
    }
}
```

Because the ConceptBridge implementation assumes a special relevance, it deserves a closer insight, especially the createTargetInstance and the runPropertyBridges methods.

The createTargetInstance method corresponds to the following code:

```
public boolean createTargetInstance(
    TransformationInformationTable TI2,
    Instance instance) throws Exception
{
    CopyInstance mafraService = new CopyInstance();
    mafraService.transformation(TI2, instance, getArguments());
}
```

It creates an object of the CopyInstance Service and calls its transformation method, which is responsible for the query-filtering-transformation-filtering-instantiation process, according to the input instance and the arguments of the ConceptBridge, including cardinality and extensional specification information. Change will be reported in the transformation information table.

The runPropertyBridges method corresponds to the following Java code:

```
public boolean runPropertyBridges (
    Engine engine,
    TransformationInformation TI ) throws Exception
{
    Set setAllPropertyBridges = getAllPropertyBridges();
    Iterator it = setAllPropertyBridges.iterator();
}
```



```
for(; it.hasNext(); ) {
    PropertyBridge pb = (PropertyBridge) it.next();
    pb.transformation(ti);
}
Set setAllAlternativeBridges =
    getAllAlternativeBridgesOfPropertyBridges();
Iterator it = setAllAlternativeBridges.iterator();
for(; it.hasNext(); ) {
    AlternativeBridgeOfPropertyBridges altPBs =
        (AlternativeBridgeOfPropertyBridges) it.next();
    altPBs.runUntilSuccessful(ti)
}
}
```

The `runUntilSuccessful` method of the `AlternativeBridgeOfPropertyBridges` class will call the `transformation` method of every `PropertyBridges` until one is successfully executed.

The `transformation` method of the `PropertyBridge` class is similar to the `createTargetInstance` method of the `ConceptBridge` class. It is defined as follows:

```
public boolean transformation(TransformationInformation ti)
{
    Class mafraServiceClass = Class.forName( getServiceLocation() );
    Class[] args = new Class[] {};
    Constructor constructor = mafraServiceClass.getConstructor(args);
    mafraService = (MAFRAService) constructor.newInstance();

    mafraService.transformation(ti, getArgumentValues());
}
```

The four first lines of previous function are responsible for the creation of an object of the `PropertyBridge` Service. This is different from the `CopyInstance` Service call because these are not built-in Services as `CopyInstance` is. Therefore, it is necessary to locate and load the code before operation. The last line calls the `transformation` method mandatorily existent in every Service.

8.1.3.4 Services

Services are responsible for the transformation of source ontology instances into target ontology instances. As proposed in Chapter 7, Services are independent, pluggable transformation modules. It means, Services do not belong to a specific MAFRA module, but provides services to many of these modules, depending on the interface defined by the MAFRA module and those implemented by the Service.

Services are intended to be added, changed or removed from the system easily and efficiently. In the scope of MAFRA Toolkit, both the object-oriented and the interface constructs available in the Java programming language have been exploited to support the intended functioning. Moreover, these two modeling constructs (i.e. object and interface oriented programming) provide the basics elements to support a fast and reliable deployment of Services.

The functionalities required by MAFRA core modules from Services are specified through the MAFRAService interface. Every Service intending to provide transformation, automatic bridging or any other functionality to the MAFRA Toolkit system must implement this interface.

MAFRA Toolkit implementation provides basic functionalities for the most of the methods defined in MAFRAService interface. In most of the cases, abstract methods are implemented in the MAFRAAbstractService, from which all specific Services can derive from, thus profiting from elementary functionalities already implemented.

Figure 8.2 represents the most important classes and interfaces of the MAFRA Toolkit Service-oriented Architecture.

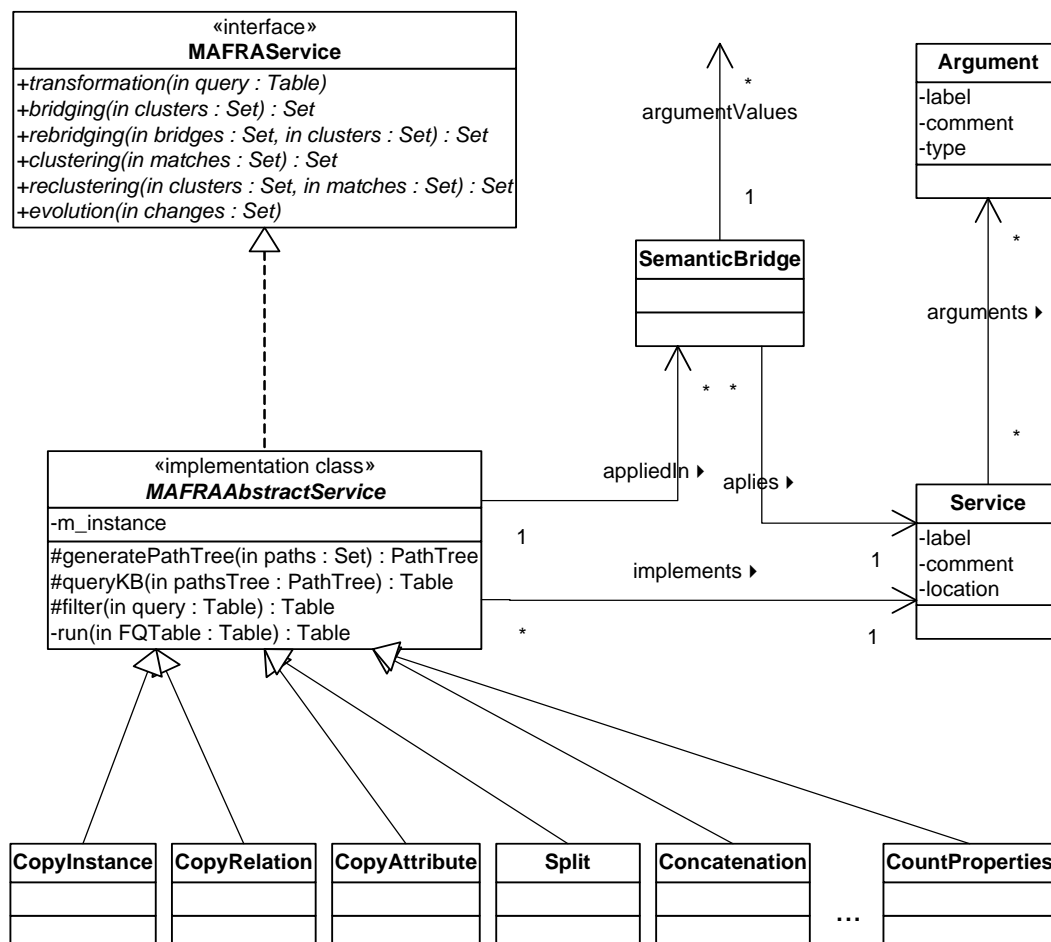


Figure 8.2 – UML representation of core classes and interfaces of MAFRA Toolkit

In particular, to support the Execution phase, any specific Service (e.g. CopyInstance, CopyRelation and Concatenation) is implemented by sub-classing the MAFRAAbstractService class, and implementing run method. The run method receives the *FQ* table and returns a table corresponding to the target entities instances to create (the last *FQ* table). The transformation method implementation provided by the MAFRAAbstractService provides the necessary functionalities in generating the *FQ* table from source knowledge base. The run method, implemented in each specific Service, provides only the specific know how in transforming the *FQ* table rows into the target instances table, whose columns reflect the name of the target entities. The MAFRAAbstractService is thereafter responsible for filtering the target instances according to the target ConditionExpressions and instantiate the remaining instances in the target knowledge base.

It is therefore considerably simple to define, implement and plug in new Services into MAFRA Toolkit.

8.1.4 Graphical User Interface

The user interface provides the representation and the manipulation facilities of the ontology and of the ontology mapping entities. Generically, ontologies and SBO mapping document can be very difficult to understand and manipulate by non-expert users.

8.1.4.1 Tree-based user interface

The representation and exploration of ontologies has been, either directly or indirectly, the focus of many research projects such as Protégé [Noy *et al.*, 2000], OntoEdit [Sure *et al.*, 2002], OilEd [Bechhofer *et al.*, 2001], WebODE [Arpírez *et al.*, 2001] and KAON [Motik *et al.*, 2003], for several years.

Figure 8.3 presents a screenshot of the KAON SOEP tree-based interface for ontology representation and development. In this figure, two ontologies are being manipulated at same time. The lowest part of each internal frame is used to represent the details of the entity selected in the upper parts. In the left ontology, properties values of the Bruce_Croft instance are being presented while in the right ontology, the attributes of the Mitarbeiter concept are presented.

Implementation of the MAFRA Toolkit user interface started under the scope of this representation paradigm. First MAFRA Toolkit GUI adopted the tree-based representation of ontologies, but developed a new interface system where two ontologies are loaded into the same frame, separated by the representation and manipulator of the ontology mapping document.

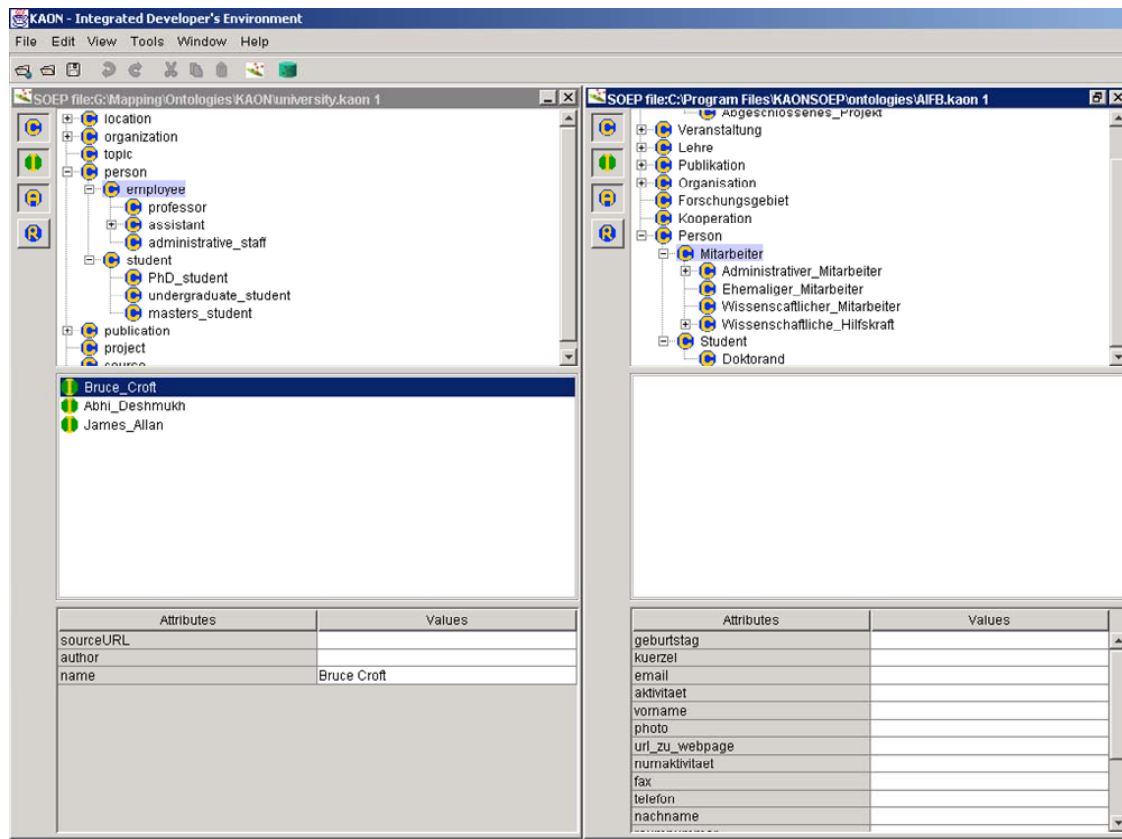


Figure 8.3 – Screenshot of the KAON SOEP tree-based interface

Moreover, all ontology deployment functionalities have been discarded and the interface focused in those of the ontology mapping process. Figure 8.4 presents a screenshot of the first implemented interface.

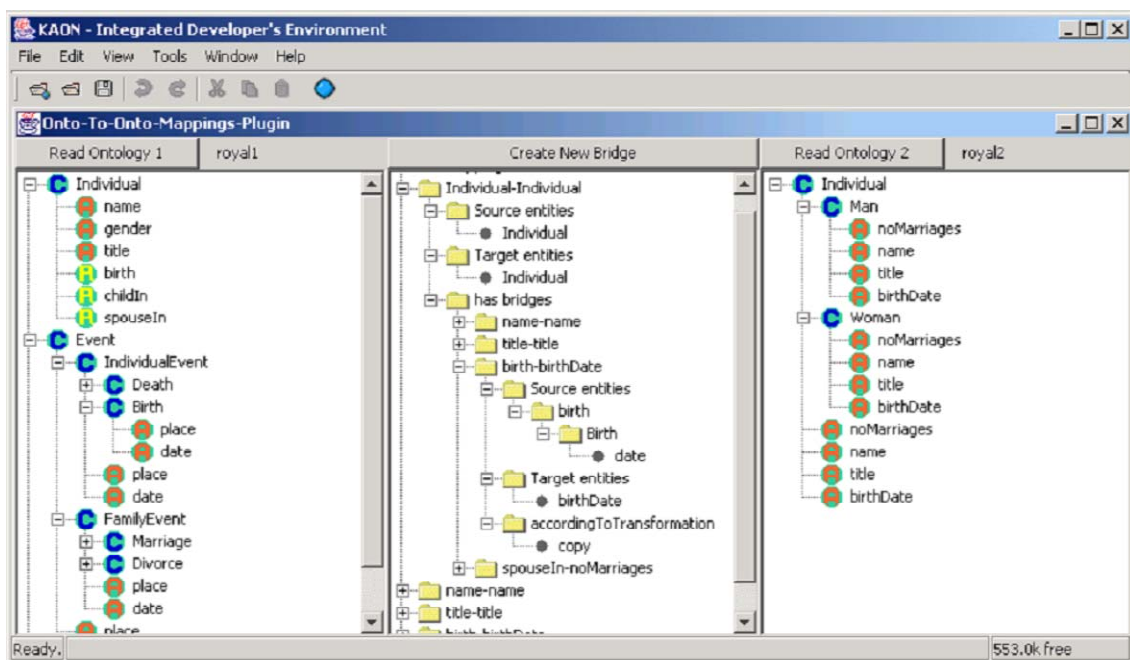


Figure 8.4 – MAFRA Toolkit UI: first tree-based implemented interface

As the ontology representation, SemanticBridges are also represented as tree-based structures, in which every branch (or sub-branch) reflects a distinct component or set of components of the SemanticBridge. In order to apply ontology entities (Concepts and Paths) the user selects the ontology entities, drags and drops them into the intended SemanticBridge argument.

As in any tree-based representation, in order to represent multiple relationships for the same entity, the entity must be represented more than once, causing ambiguity in the interface. Moreover, this type of representation of the mapping document becomes too extensive and thus hard to understand and manipulate. In order to cope with this problem, a simple evolution was made in the GUI such the central frame represents only the SemanticBridges and their inter-relations (i.e. \prec relation between ConceptBridges, the \diamond relation between ConceptBridges and PropertyBridges, \perp^{B^C} relation between ConceptBridges and AlternativeBridges-of-ConceptBridges and \perp^{B^P} relation between PropertyBridges and AlternativeBridges-of-PropertyBridges).

The other SemanticBridges properties are represented and manipulated in a new frame of the GUI, referred as Details Panel. This frame is presented in the lower part of the user interface. The information presented in the Details Panel comes from the SBO entity instance selected in the upper central panel. Figure 8.5 corresponds to a screenshot of MAFRA Toolkit user interface conforming to the described stage of evolution.

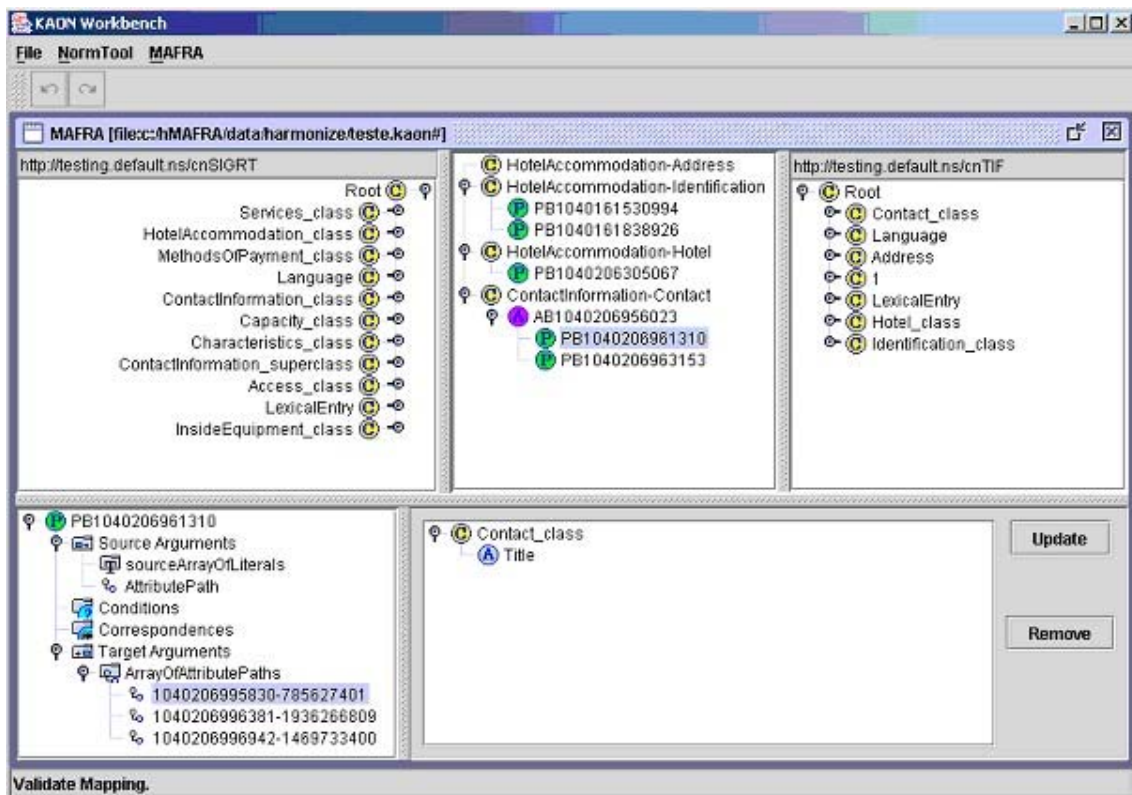


Figure 8.5 – MAFRA Toolkit UI: distinct panels for SemanticBridges and their parameters

Tree-based interfaces have been used to represent the hierarchical structure of ontologies for some time, but it became clear that such representation did fit neither the property-centric modeling approach nor the network structure resulting from it. Moreover, tree-based interfaces were far from good ontology manipulation tools, especially when dealing with large ontologies, which starts to be frequent nowadays.

8.1.4.2 Net-based user interface

KAON infrastructure and its ontology development editor evolved in a way that ontologies are no longer represented as tree-based structures but as a network of multi-inter-related entities. This new representation approach allows the representation of multiple relations types for the same entity with no need for multiple representations of the entity. While this new representation approach has been applied to the ontology editor, functionalities were made available as a library of classes to other (eventual) KAON components. This library has been exploited in the development of a new, more usable and flexible ontology mapping editor. Figure 8.6 represents a screenshot of the user interface of MAFRA Toolkit comprehending these last changes.

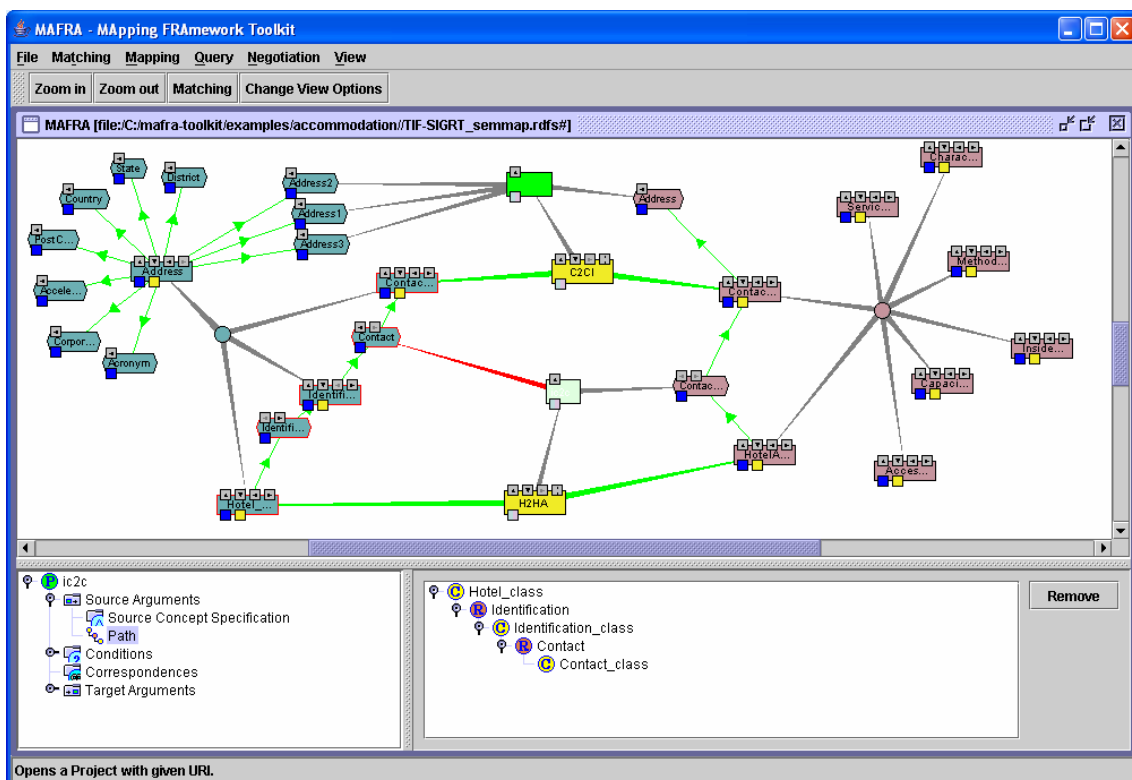


Figure 8.6 – MAFRA Toolkit UI: net-based representation of entities

The result is a new interface in which entities could be selected, hide, deleted, moved, drag and dropped, related and unrelated in a very versatile and rather intuitive fashion.

In this new representation approach, two basic types of entities exist:

- The node, which typically represents the ontology or mapping entities and their instances;
- The edge, which typically represents the ontological and mapping relations between their entities.

For every type of ontology, mapping entity or instance, a distinct shape and color representation can be used, providing a very perceptible representation of the mapping scenario. Distinct shapes and color are also used in edges according to the relationship it represents. Table 8.1 presents some of the most used entities according to their shape and color:

Table 8.1 – Shape-color characterization of mapping entities

Entity	Shape	Color	Example
Ontology	small circle	dark green/maroon ⁵²	
Concept	rectangle	dark green/maroon	
Property	irregular hexagon	dark green/maroon	
ConceptBridge	rectangle	yellow	
PropertyBridge	rectangle	light green	
AlternativeBridge	rectangle	light blue	
Matches	small square	dark blue	
Domain/range relation	line with arrow	light green	
Entity to ontology relation	pointed line	gray	
Concept to ConceptBridge relation	pointed line	light green	
Path to PropertyBridge relation	pointed line	gray	
< (subBridgeOf) relation	pointed line	gray	

All manipulated entities can be simultaneously represented in the user interface, which represents an important improvement comparing to the tree-based user-interface. For example, even if Matches and SemanticBridges are entities associated with distinct stages of the ontology mapping process, in some moment their manipulation occurs at the same time, and should therefore be represented and manipulated simultaneously.

⁵² The source ontology is represented in dark green and target ontology is represented in maroon. The same colors are applied in representing either source or target ontologies' entities.

Figure 8.7 is a screenshot of the MAFRA Toolkit, in which many types of entities are represented, including a set of Matches and the SemanticBridges automatically generated from them.

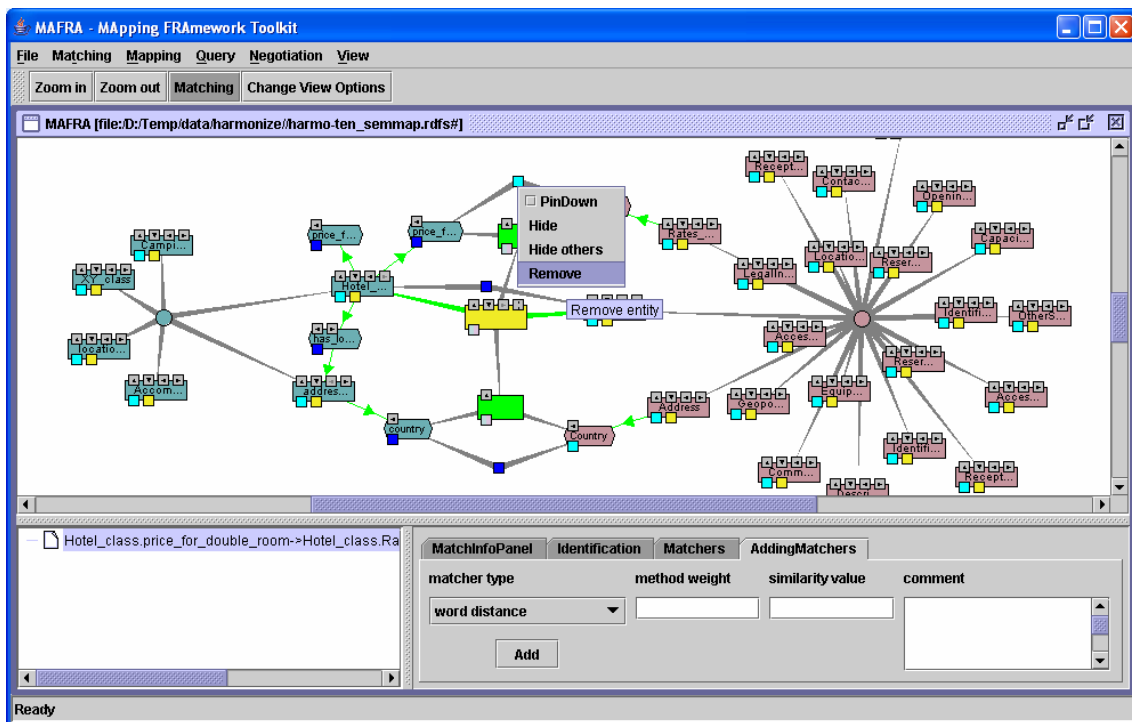






Figure 8.7 – MAFRA Toolkit UI: simultaneous representation of all types of entities

Besides the entity actions accessed through the context menu, most of the node-represented entities are also manipulated through the node-represented buttons. Like any interface button, these are also a two state buttons. In these cases, buttons correspond to collapse and expand of relations of a certain type (e.g. subBridgeOf relation between ConceptBridges). Table 8.2 represents the semantics of each button in context of each node-represented entity:





Table 8.2 – Semantics of every button available in node-represented entities

Button	Node	Semantics
◀	Concept	Expand/collapse properties that have this concept as range.
	Property	Expand /collapse domain concepts of this property.
▶	Concept	Expand/collapse properties that have this concept as domain.
	Property	Expand/collapse range concepts.
	ConceptBridge	Expand/collapse \diamond -related PropertyBridges or AlternativeBridges-of-PropertyBridge.
◻	AlternativeBridge	Expand/collapse \perp^{B^c} or \perp^{B^p} -related Concept or Property Bridges.
	Concept	Expand/collapse sub-concepts.
◻	ConceptBridge	Expand/collapse \prec -related ConceptBridges (sub-bridges).
	Concept	Expand/collapse sub-concepts.
◻	Concept	Expand/collapse sub-concepts.
	ConceptBridge	Expand/collapse \prec -related ConceptBridges (super-bridges).

	ConceptBridge	Expand/collapse AlternativeBridges-of-ConceptBridges to which this ConceptBridge is \perp^{B^c} -related.
	Concept	Expand/collapse Matches to which this Concept is related.
	Property	Expand/collapse Matches to which this Property is related.
	Concept	Expand/collapse ConceptBridge to which this Concept is related.
	ConceptBridge	Expand/collapse Concepts that this ConceptBridge relates.

Every button represents their expanded/collapsed state by changing its dark color (when collapsed) to a light color (when expanded). However, the state of the button does not guarantee the state of the entity/relation. This is due to the fact that the relations can be manipulated in both sides of the edge.

Example 8.3 – Ambiguous appearance of graphical buttons

Expanding the  button of a Concept, all properties of that Concept are shown and an edge is created between the Concept and every Property. The button color changes to gray (). If the  button of Properties is pushed, the previously expanded edges are collapsed, but the button of the Concept is maintained gray ().

A few other features provide interesting and useful functionalities to the user:

- Selecting an entity changes its color to a lighter color (e.g. Matches change from dark blue to light blue), and triggers the presentation of other characteristics of the entity in the lower frame of the user interface. In Figure 8.6, the ic2c PropertyBridge is currently selected;
- Context menu is available for every entity, varying according to the selected entity and allowed commands. In Figure 8.7, the context menu is presented for the Match between price_for_double_room and EuroMaximum;
- Zoom in/out permits to enlarge/reduce the representation of the entities. This functionality is important especially dealing with large ontologies, allowing focusing or generalizing the view upon the entities;
- Versatile positioning of the entities in the graphic. Three types of positioning are available:
 - Automatic positioning, in which the positioning system is completely responsible for the location of every entity, and for the relative relations between entities;
 - User defined positioning allows user to move the entity to any location, which in turn moves other entities as well, due to their relationships and location constraints;
 - Fixed positioning allows user to define the position of the entity, which will be kept independently of the changes in the graphic (add, remove, move, etc. of entities). The PinDown command accessible from the context menu (Figure 8.7) permits to fix or unfix the entity location;

- Hide feature permits to remove certain entity from the interface, while maintaining it in the ontology or mapping. While possible, this feature is unpractical and inefficient when using the tree-based representation.

The presented net-based user interface represents a great improvement to the user-based Semantic Bridging phase in comparison to the tree-based representation. In particular, it became possible to define and manipulate backward Paths; something that was almost impossible with the tree-based representation of ontologies.

8.1.5 Outlook

While its implementation is currently and continuously running, MAFRA Toolkit is already an effective tool supporting the lift & normalization, similarity measuring, (automatic) semantic bridging and execution phases of the ontology mapping process, as have been described in prior chapters. Yet, many improvements are currently being pursued as described in Chapter 10.

8.2 Application experiences

While MAFRA Toolkit implements a set of new research ideas, its application in several third-part research projects and academic curricula demonstrates, to some extent, its pragmatic merit, along with the scientific relevance.

The work described in this thesis has been developed firstly in the scope of the SANSKI - Semi-Automatic Negotiation Service for Knowledge Interoperability project [SANSKI]. SANSKI is funded by the Portuguese Technology and Science Foundation (Fundação para a Ciência e Tecnologia) (reference POCTI/GES/41830/2001) and runs from January 2002 to January of 2005.

SANSKI aims to research and develop solutions on supporting proactive interoperability between socio-economic entities represented by agent-based systems, especially in the application domains of manufacturing systems and virtual enterprises. With the increased weight of Semantic Web as application domain and technology repository, SANSKI research evolved into this domain too, and adopted some of the provided technology.

Because the SANSKI proposal suggested development of experiences and tests in real-world scenarios, a considerable effort has been putted in pursuing these goals. However, as often referred during the thesis, due to the intrinsically ambiguous nature of the problem, the solution apparently suited for a set of specific interoperability scenarios is not recommended for others. Due to the diversity of scenarios required, it has been considered unfeasible to develop in-house sufficient experiences to demonstrate the validity and usefulness of the research ideas proposed.

In response to this problem, a set of research policies have been setup that strongly contributed to the results of this thesis:

- Publicize research in high-ranked research events and publications from distinct domains of research and application;
- Develop a network of research collaboration that could provide valuable and specialized suggestions in different areas of research and application;
- Set the developed software packages publicly available, and promote its application on third-part projects and research institutions;
- Use of standard but specialized software components in MAFRA Toolkit.

As a result of these policies, the research ideas and MAFRA Toolkit have been extensively requested for application and further development by EU-funded projects and business organizations. Notice however that despite the application of MAFRA research and MAFRA Toolkit as has been supported and promoted by the author, no official or institutional relation exists between author and research groups responsible for the application in mentioned projects. Thus, the application of MAFRA Toolkit has been an exclusive decision of the research groups involved in these projects. Some of these applications and experiences are described in next sections.

8.2.1 Harmonise and Harmo-TEN

The MAFRA Toolkit has been adopted as the specification, representation and transformation mechanism in the EU-funded Harmonise project (IST 2000-29329) [Harmonise]. The Harmonise project intends to overcome the interoperability problems occurring between major tourism operators in Europe. Problems arise due to the use of distinct information representation languages like XML and RDF, and different business and information data models, like those provided by MEK [MEK], WhatsOnWhen (WoW)⁵³ [WoW], TIS⁵⁴ [TIS], SIGRT [SIGRT] and TourinFrance [TourinFrance]. Harmonise adopted a mediation approach based on the so called Interoperability Minimum Harmonisation Ontology (IMHO) which serves as *lingua franca* between entities. Every entity in the system however, does not directly interoperate with business partner, but with the mediator who transforms messages according to the predefined ontology mapping documents between IMHO and each of the entities ontologies. Contents of the messages are then wrapped and forwarded to the respective business partners conforming to the format and semantics of the receiver.

⁵³ WhatsOnWhen is an English company expert in providing event information and ticketing.

⁵⁴ tiscover A.G. is an Austrian company and the major European tourism information system provider that maintains a database of tourism data of several countries.

The IMHO describes the Accommodation and Event partitions of the tourism domain only. Despite IMHO reflects extensive efforts in modeling those partitions considering the existent ontologies, they differ very much. In fact, each of them differ so much from all the others that it would be impossible to create an ontology that resembles simultaneously some of them. Based on the type and number of entities defined in ontologies, Table 8.3 depicts the schematic differences between IMHO and other ontologies applied in Harmonise, according to the domain of knowledge.

Table 8.3 – Comparison between ontologies according to domain of knowledge

Ontology	Accommodation			Event		
	Concepts	Properties		Concepts	Properties	
		Attributes	Relations		Attributes	Relations
IMHO	136	340	543	86	174	328
WoW	-	-	-	20	40	20
MEK	2	104	1	1	47	0
TIS	26	57	26	38	41	38

The differences between ontologies are evident, but in most cases IMHO always represents a superset of the knowledge represented in the other ontologies or it applies a finer grained conceptualization, which motivates the substantial use of the extensional specification mechanism.

These enormous differences have consequences at ontology mapping document. In fact, in most of the ontology mapping scenarios presented in Table 8.4 and Table 8.5, only a small subset of the ontologies entities are semantically related.

Table 8.4 – Ontology mapping experiences in the Event domain

Source Ontology	Target Ontology	ConceptBridges	PropertyBridges
IMHO	WoW	12	33
WoW	IMHO	26	64
IMHO	MEK	1	39
MEK	IMHO	28	70
TIS	IMHO	23	46

Table 8.5 – Ontology mapping experiences in the Accommodation domain

Source Ontology	Target Ontology	ConceptBridges	PropertyBridges
MEK	IMHO	26	74
IMHO	TIS	11	45

However, as referred by the research team responsible for the ontology mapping specification, this fact was only a concern of ontology mapping decisions and not a limitation of the ontology mapping tool. In fact, no conceptual limitations have been detected in MAFRA Toolkit, and only a

few Services have been developed as refinement of the initially provided Services (e.g. Split by regular expression Service has been developed from the original Split Service).

Harmonise is a successfully completed research project that formally ended in July 2003. Yet, a follow-up has been recently approved by the European Commission's eTEN-Programme (eTEN C510828) under the name of Harmo-TEN [Harmo-TEN]. In this new stage, the goal of the project is the market validation of the business concept and services of the Tourism Harmonisation Network (THN), created by Harmonise. MAFRA Toolkit has already been adopted as background solution in the project, which means that it will be applied as the specification, representation and transformation component of the mediation system. Due to this fact, it can be easily inferred that MAFRA Toolkit and especially the research ideas it implements fulfilled requirements found in previous stage, and it is envisaged as solution in this new, more demanding stage.

8.2.2 Artemis and Satine

Currently, MAFRA Toolkit is being applied in the Artemis (IST-1-002103-STP) [Artemis] and Satine (IST-2104) [Harmo-TEN; Satine] EU-funded projects.

On one hand, Artemis aims to develop a system that allows and promotes the discovery and exchange of healthcare information existent in repositories conforming to distinct representation standards [Harmo-TEN; Laleci *et al.*, 2004].

Satine, on the other hand, aims to develop a system that promotes interoperability of small and medium enterprises in the tourism domain [Sinir *et al.*, 2004].

Although very different in application domains, both projects adopt a very similar technological approach:

- Both rely on Web Services to provide the interoperability process;
- Both envisage the combination of Web Services of different granularities to provide a general-purpose information interoperability system;
- Both rely on ontologies derived from information standards used in healthcare and tourism domains to classify and publish the competencies of web services;
- Both rely on ontology mapping process, and in particular in MAFRA Toolkit to transform documents represented in distinct information standards.

According to authors, four characteristics have been considered for the application of MAFRA Toolkit in both projects:

- Ability to manage ontologies as ontologies and not as tree-based documents;
- Easy specification, definition and representation of semantic relations;

- Extensibility of the system, namely respecting the fact that Services are easily developed and pluggable into the system;
- Open code and publicly available.

While none of the projects have made public any ontology mapping document yet, it is possible to envisage many ontology mapping scenarios. In the healthcare domain for example, multiple information standards are used upon the same domain (e.g. HL7⁵⁵, CEN TC251⁵⁶, ISO TC215⁵⁷ and GEHR⁵⁸), motivating the specification of semantic relations between them, in order to promote interoperability between repositories.

The healthcare domain is particularly attractive for the MAFRA Toolkit, especially due to the required level of accuracy in the manipulation of medical information. In fact, MAFRA Toolkit has not been applied in scenarios where such level of accuracy is so important and decisive for the adoption of the system itself.

8.2.3 BRIDGE-IT

Probably the most interesting and extensive third-part reference to this research has been done in the scope of the EU-funded project BRIDGE-IT (IST-2001-34386). The BRIDGE-IT (Bringing Innovative Developments for Geographic Information Technology) aims to close the gap between research products and ready-to-use products in the field of the GIS. In its Technology Watch Report 4 [Janowicz & Riedman, 2004], the team from University of Munster, Germany, overviews, analyzes and comments this research work as “a very good example for the ongoing work done in the area of ontology mapping”. Authors defend the relevance and usefulness of MAFRA Toolkit and the research it proposes but, at same time, probably because of the scope of the BRIDGE-IT project (to promote research into ready-to-use products), authors suggest the need for a more definitive implementation.

⁵⁵ Health Level 7 (HL7), <http://www.hl7.org>

⁵⁶ CEN TC/251 (European Standardization of Health Informatics) ENV 13606, Electronic Health Record Communication, <http://www.cen251.org/>

⁵⁷ ISO TC/215, International Organization for Standardization, Health Informatics Technical Committee, <http://www.iso.ch/iso/en/stdsdevelopmenttc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=4720>

⁵⁸ The Good Electronic Health Record, <http://www.gehr.org>

8.2.4 Outlook

Several applications of MAFRA Toolkit have been referred in this section. While the decision on applying MAFRA Toolkit is exclusive of the research team involved in the project, technological support has been provided by the author.

This is especially true concerning the application of MAFRA Toolkit in Harmonise project. In fact, due to the fact that MAFRA Toolkit has been adopted in early phases of its own development, in many periods of development, the feedback and requests for functionalities was very large, which could have motivated some development dependency. However, since then, many other projects start applying MAFRA Toolkit and the research ideas it preconizes, which is understood as sign that the co-operation with Harmonise did not affected the general-purpose and advanced semantic bridging capabilities envisaged for this research work. This sign is further confirmed with the use of MAFRA Toolkit in academic contexts. It seems therefore that the benefices of co-operation with third-part projects are larger than the disadvantages.

8.3 Performance and comparison experiences

Performance is often an important topic when describing research ideas in this research field. In the case of ontology mapping systems and especially concerning the MAFRA Toolkit, performance may be related to at least two components of the system:

- Execution process, concerning the number of instances transformation per unit of time;
- Automatic semantic bridging process, concerning the time necessary to create the SemanticBridges according to the number of the entities of the ontologies.

Concerning the automatic semantic bridging process, no performance experiences have been carried out. This is especially due to the fact that in current stage of the research ideas and developed system, reduction of ambiguity and improvement of accuracy of the automatically proposed SemanticBridges are the fundamental subjects the research is concerned with. Consequently, it makes no sense for the moment to carry out performance experience upon this process.

On the other hand, some experiences have been carried out concerning the performance of the execution process. These experiences though, have been strongly influenced by the nature of the problem and research context. Formal performance experiences are normally carried out through the execution of a battery of tests that is used by the research community. The comparison tests serve often not only to judge the performance capabilities of the system but also other dimensions of the problem, namely the quality of the results.

Yet, in current research context, such experiences are unable to be performed. In fact, the research community concerned about the ontology mapping problem is recent and insipidly established, thus lacking some fundamental research elements such is the battery of case tests and comparison reports.

Considering the lack of comparison reports between ontology mapping tools, the performance experience reported by Dou and colleagues, first in [Dou *et al.*, 2002] and later in [Dou *et al.*, 2003] constitute a simple but valuable piece of work. The ontology mapping system applied in these report is OntoMerge, which has been described in 5.3.4. OntoMerge and MAFRA Toolkit are very different as is demonstrated in Table 8.6:

Table 8.6 – Some differences between OntoMerge and MAFRA Toolkit

Characteristics	OntoMerge	MAFRA Toolkit
Semantic bridging strategy	mapping through merging	(pure) mapping
Execution strategy	inference-based	functional
Semantic relations representation	axioms in Web-PDDL	RDF instances of SBO
Ontology representation	Web-PDDL	RDFS with lexical extensions
KBs representation	Web-PDDL	RDF
Input	RDF file in the Web	RDF file
Output	RDF file in the Web	RDF file
Execution environment	through the Web	Java virtual machine
Development availability	private	public

The difference between the report in [Dou *et al.*, 2002] and that presented in [Dou *et al.*, 2003] concern the time required in different stages of development of the OntoMerge system to perform the same knowledge base transformation, and according to the same bridging axioms.

In both publications authors report the experience in mapping two ontologies about genealogy:

- The Gedcom ontology [Gedcom];
- The Gentology ontology [Gentology].

Because these are very similar ontologies, the resulting ontology mapping scenario requires only simple semantic relations. In this sense, the comparisons presented so forth should be understood as performance tests and not evaluation of semantic bridging capabilities. Concerning this last issue, the experiences made in third-part projects are more informative and relevant.

Because OntoMerge and MAFRA Toolkit are very different, it has been necessary to specify in the MAFRA Toolkit the semantic bridges of the bridging axioms of OntoMerge. The specified ontology mapping document using MAFRA Toolkit specifies the semantic relations presented in the previously mentioned reports and from examples found in the OntoMerge web pages.

Once the ontology mapping document is finished, the execution experiences took place. The knowledge base applied in the reported transformation experiences is composed of 21164 instances (facts) about the European royalty. No differences have been detected between the target instances resulted from transformations executed by OntoMerge and MAFRA Toolkit.

The experiences with OntoMerge have been reported in two distinct publications, in which two completely distinct execution times have been referred:

- According to [Dou *et al.*, 2002], OntoMerge requires 22 minutes execution time in a Pentium III at 800MHz;
- Later, in [Dou *et al.*, 2003], after non-described improvements, it has been reported that OntoMerge executed the same transformation in 59 seconds.

Unlike experiences with OntoMerge, MAFRA Toolkit experiences have been performed for the same publication [Silva & Rocha, 2003e], and the same original implementation of MAFRA Toolkit has been used. Instead, the tests ran in two distinct machines, achieving distinct results:

- In a Pentium II at 350 MHz the MAFRA Toolkit took less than 2 minutes;
- In Pentium 4M at 2.0Mhz the MAFRA Toolkit required less than 77 seconds.

During the tests performed with MAFRA Toolkit it has been impossible to determine an exact duration of the transformation process. In the last scenario, the minimum time necessary has been 64 seconds while the maximum has been 77 seconds. Similar behavior has been observed when using the Pentium II-based machine. Table 8.7 summarizes both experiences.

Table 8.7 – Comparison summary of performance experiences

OntoMerge		MAFRA Toolkit	
Pentium III 800 MHz		Pentium II	Pentium 4M
2002 experience	2003 experience	2003 experiences	
1320 seconds	59 seconds	< 120 seconds	< 77 seconds

Even if only a small difference exists between best performances of both systems, the fact is that according to these experiences OntoMerge system performs better than MAFRA Toolkit.

Although no definite reason can explain these results, there are a few facts that should be referred:

- No performance concerns have been considered when developing MAFRA Toolkit;
- No query optimized libraries or methods have been adopted in MAFRA Toolkit. Instead, only the original KAON API provided for the (simple) access of ontologies and knowledge bases has been used. As consequence, all query and filtering methods described in Chapter 6 have been implemented recurring to general-purpose data structures;
- No re-engineering of code has ever been done. This is particular relevant because:
 - The implementation of the execution process has occurred during the specification phase,

which typically motivates bugs and performance problems;

- OntoMerge suffered modifications that resulted in a performance improvement of 2600%. Even if it is not plausible that such improvement can occur in MAFRA Toolkit, it is expectable that some improvement can be achieved;

Yet, the achieved results are satisfactory enough in current research context, especially because focus has been set on the semantic bridging and execution capabilities of the research/system and not in performance issues.

8.4 Conclusion

In this chapter, the development, application and comparison of MAFRA Toolkit have been described. Due to MAFRA Toolkit, the research ideas presented in this thesis have been exposed to a large community of researchers, academics and business enterprises as would not be possible otherwise.

According to the applications and experiences made by third-part projects and taking in consideration their opinions, MAFRA Toolkit is:

- Useful, because it has been applied by many third-part projects;
- General-purpose, since it has been applied in very distinct application scenarios;
- Valid, because it has been successfully applied in those projects and scenarios;
- Scientifically relevant, due to its application and development by third-part research groups.

However, despite the visibility it provided, MAFRA Toolkit permitted to receive feedback that will be helpful in future research and development stages. Some of the most relevant suggestions received by the practitioners are presented in Chapter 10.

However, it is already clear that one of the most sensitive components of the work executed during this thesis is the MAFRA Toolkit. In fact, performance, code structure, GUI functionalities and stability may be largely improved by re-engineering MAFRA Toolkit code.

THIRD PART

Chapter 9

CONCLUSION

This chapter presents an overview of the research and development work described in this thesis. Due to the intrinsically subjective nature of the ontology mapping problem, a formal conclusion on the applicability of the proposed research ideas cannot be drawn. Yet, the application of the MAFRA Toolkit by many third-part projects worldwide provided a useful and supposedly correct impression on the competencies and limitations of the proposed research ideas and of MAFRA Toolkit itself.

Thus, even if a formal conclusion is conceptually difficult in this context, it is possible to enumerate the main contributions of this thesis followed by an enumeration of the most relevant achievements of this thesis according to the research community.

9.1 Outlook of the thesis

The work described in this thesis has four main components:

- The contextualization and motivations for the research presented in this thesis have been described in Chapter 1, Chapter 2 and Chapter 3;
- The theoretical research part, in which the most relevant research ideas and approaches proposed in the scope of this thesis have been extensively presented and analyzed, corresponds to Chapter 4 through Chapter 7;
- The implementation and application experiences, in which the MAFRA Toolkit and its applications have been described corresponds to Chapter 8;
- Conclusion and future research are described in Chapter 9 and Chapter 10.

Next sections summarize each of these components.

9.1.1 Contextualization and motivations

First chapter of this thesis described the background research context of this thesis. Previous work in the area of agent-based manufacturing systems demonstrated the need to provide the agent-based entities (i.e. cooperative, proactive) with the abilities to engage in conversations with other entities from distinct information communities. Knowledge-based interoperability between entities arises as one of the main requirements of socio-organizational systems, since it permits the dynamic and emerging configuration of conversations and business processes. In this context, the ontology-based representation of interoperability information and its exploitation by the ontology mapping process had risen as one of the most promising solutions to overcome the interoperability gap between information communities.

In Chapter 2, a set of similar problems have been identified in different technological domains, in which the same potential solution would be advantageous and applicable. Describing and analyzing such scenarios, a set of common relevant requirements have been systematized. These requirements have been further applied as research goals throughout the research and development work.

Finally in Chapter 3 the notion of ontology has been described and analyzed according to a commonly referred set of meaningful characteristics. This analysis has been further applied in the comparison of ontology with the concept of database model. Later, the concept of ontology has been formally defined, serving as the central and univocal notion of ontology during the rest of the thesis.

9.1.2 Theoretical research

Once contextualized and several constraints defined, the research efforts started. In Chapter 4 the MAFRA – MApping FRAmework has been presented. MAFRA is the first and only known analysis and systematization of the ontology mapping problem. It represents a generic but rather complete perspective of the overall ontology mapping process, not only under the point of view of the fundamental process phases, but also considering the complementary tasks and components.

The two fundamental phases of the ontology mapping process have been described in Chapter 5 and Chapter 6.

The semantic bridging phase, described in Chapter 4, consists in the specification of semantic relations between source and target ontologies entities. Because no sufficiently featured representation language was found, a specific semantic relation representation language has been specified and developed: SBO. SBO is an ontology of the semantic relations as perceived in the ontology mapping domain of knowledge. An SBO instantiation represents the semantic relations holding between two ontologies under the perspective of a specific user/domain expert.

SBO has been extensively and formally described, providing a univocal understanding of the semantic relations and of their inter-relations, which permitted the partial automation of the semantic bridging process, as described in Chapter 7. Due to its formal specification, multiple notations and syntaxes can be used to represent ontology mapping documents. However, in the scope of this thesis SBO has been only partially specified in ontology representation languages. This is due to the limited expressive power of the Semantic Web aware ontology representation languages which are clearly insufficient to represent all rules and constraints defined for SBO. Yet, SBO has been programmatically represented in MAFRA Toolkit, in which an interpreter and validator of SBO have been implemented.

The Semantic Bridging Ontology and consequently the execution process (described in Chapter 6) adopted a new ontology mapping strategy, by combining rule-based transformation with Description Logic-like specification of semantic relations. The rule-based approach allows the declarative specification of semantic relations, providing an intuitive and immediate mapping specification. The DL specification is applied to overcome conceptual mismatches between ontologies, namely concerning the heterogeneity produced by distinct granularity. Furthermore, the modular structure of SBO, in which each ontology concept is semantically bridged independently of others concepts, is perfectly suited to facilitate the evolution of the ontology mapping document according to ontologies changes.

Besides its importance on the process, the specification and representation of semantic relations is inconsequent if no transformation process occurs from source to target ontologies entities. This

transformation is accomplished on the execution phase of MAFRA – MAPPING FRAMework. The research concerning with this phase has been described in Chapter 6.

The proposed execution process is based on five distinct phases:

- Query the source knowledge base for the instances that are addressed in the Semantic Bridge;
- Filtering the source instances according to specified source conditions;
- Transformation of the source instances resulting from previous phase into target instances;
- Filtering the resulting target instances according to specified target conditions;
- Instantiation of target instance in the target knowledge base.

The query and filtering phases have been described based on the relational algebra operators, providing an explicit, formal and compact specification of the method. The deeper difficulties found concerned with the need to query the source knowledge base semantically coherent with multiple Paths. For that, a tree-based representation of the Paths has been developed, supplying the driving mechanisms for partial and incremental query of Paths. The result is a relation (table) whose attributes are the Paths specified in the SemanticBridges and the values are the instances of the knowledge base, providing the direct access to all source instances to transform before the transformation initiation. Once the query phase is finished, the filtering phase takes place according to the source ConditionExpressions defined in the SemanticBridge.

The transformation of instances is performed by special entities called Services, as generically introduced in the SBO specification. In this context, Services are responsible for receiving the source instances from the execution engine and transform them into target instances according to the specific transformation each Service represents/implements. The resulting target instances are stored in a new table such the next phase can filter them according to the target ConditionExpressions. In the fifth phase, the remaining target instances are effectively created in the target knowledge base.

The system architecture described in Chapter 7 extrapolates the notion of Service into an external, pluggable entity, competent not only in the transformation of instances, but also in other tasks related to the overall process. Due to its multiple featured nature, Services are referred as Multi-dimensional Services and are one of the main components of the proposed Multi-dimension Service-Oriented Architecture. In this architecture, Services acquire, represent and provide competencies commonly associated with domain expertise to MAFRA modules. Due to their modularity, domain expertise is modeled as multiple modules evolving and adapting independently of each others, which is apparently more suited to adapt to and support the distributed and highly dynamic paradigm of Semantic Web. Evolution, common-consensus building, validation and automatic semantic bridging are envisaged as processes benefiting from the proposed architecture.

In order to analyze and test the actual and potential contributions of the multi-dimensional service-oriented architecture, it has been decided to research and develop an automatic semantic bridging process that would apply and exploits the ideas proposed by the architecture. In fact, besides adopting the architecture proposals, the automatic semantic bridging process extrapolates the ideas to other dimensions of the MAFRA – MAPPING FRamework. In particular, the process suggests that matchers providing the similarity measures between ontologies entities are modeled as external, pluggable entities, as suggested for Service concept, by the architecture. The outcome of the matches is then used by the automatic semantic bridging process. This process pushes the similarity measures into Services, by requesting special-purpose competencies of Services, as suggested by the architecture.

9.1.3 Development and experiences

While the theoretical part of the work has the most considerable substance of this thesis, the development and experimental aspects were also very effort-demanding and time-consuming. In fact, development efforts ran parallel during most of the theoretical development time. This close relation is considered beneficial for both components. In particular it helped to:

- Prove the feasibility and usefulness of the proposed research ideas;
- Provide feedback on competencies and limitations of the research ideas;
- Promote the ideas into a larger scientific community and business audience;
- Provide a running tool to be used by third-part projects, which in turn provided a larger and consistent feedback based on experiences in a broader set of applications scenarios.

The MAFRA Toolkit is the main outcome of this part of the work. While most of the implemented functionalities concerns the ideas proposed at the theoretical part, some functionalities have not been the focus of systematized research but a matter of pragmatism. This is especially true for the GUI, in which the diverse types of ontology and ontology mapping representations developed, demonstrates this experimental pragmatism.

Yet, the implementation efforts have been strongly influenced by the decision to use the KAON Workbench as background technology for the manipulation of ontologies. Despite its competent API for ontology manipulation, KAON provides other competencies such as the ontology editor and the set of ontology evolution strategies and GUI libraries.

However, probably the most relevant limitation of KAON Workbench concerns its lack of support of an ontology-based query language. In fact, all the query and filtering processes described during Chapter 6 have been all implemented in the scope of this work, constituting a considerable work load in the development part of the thesis. Additionally, due to the considerable experience-based research adopted after a certain stage of the research, prototyping the changes in the execution

engine have been considerable difficult and time-consuming. However, the execution engine and Services are now stable and efficiently running.

9.2 Summary of research achievements

While formal conclusions cannot be drawn upon the work described in this thesis, this section aims to systematize its main contributions:

- The MAFRA - MApping FRAMework:
 - MAFRA systematizes and organizes the phases of the ontology mapping process;
 - Represents the advocated ontology mapping process;
 - MAFRA claims that the ontology mapping process, like a typical system or product is ruled according to a life-cycle perspective;
 - MAFRA advocates an iterative, interactive flow of results between phases of the process;
 - MAFRA integrates specific but essential modules related to the operationalization of the process;
 - MAFRA provides an efficient framework to classify and characterize related works from different research fields.
- The Semantic Bridging Ontology (SBO):
 - SBO defines a taxonomy of SemanticBridges representing the types of semantic relations holding between entities of two ontologies;
 - SBO defines additional concepts and inter-relations into a general-purpose yet compact and highly expressive conceptualization of the semantic relations;
 - SBO extends the transformation capabilities of the ontology mapping system by distinguishing between the set of entities semantically related and the transformation component of the relation (transformation Service);
 - SBO provides a declarative mean to specify, represent and convey the semantic relations holding between two ontologies (ontology mapping document) in a variety of notations and syntaxes.
- The Execution Process:
 - General-purpose transformation process, based on five distinct phases: query, filtering transformation, filtering and instantiation;
 - Supports transformation constraints according to both source instances and target instances resulting form the transformation;
 - Fully formal description of the process, based on the relational data model and in the relational algebra;

- Functional transformation process, supported through external transformation Services that can however be easily expanded;
- It does not use the *skolem terms* theory [Russel & Norvig, 1995] but instead to extensional specification (DL-based) of instances.
- The Multi-dimensional Service-oriented Architecture:
 - The core phases of MAFRA are operationally organized into system components;
 - The notion of Service is expanded into the so called Multi-dimensional Service, representing and embodying other competencies besides transformation;
 - Services are specified as independent, auto-characterized, external, pluggable components;
 - Services competencies are requested by core modules through the MAFRA Service Interface, providing functionalities according to each specific process phases.
- The Automatic Semantic Bridging Process:
 - Developed as a case test of the Multi-dimensional Service-oriented Architecture;
 - Derives a valid ontology mapping document between two ontologies;
 - Applies a variety of similarity measures (matches) between ontology entities in the process, evaluated by a variety of independent, external, pluggable components named Matchers;
 - Expand Services competencies useful in the automation of the semantic bridging phase;
 - Services define the conditions (based on matches) that must hold in order to be associated in a semantic relation between a set of source and target ontologies entities;
 - Adopts, follows and explores the conceptualization proposed by SBO.
- The MAFRA Toolkit:
 - Implements the proposed Multi-dimensional Service-oriented Architecture;
 - Implements an interpreter and validator of SBO ontology mapping documents;
 - Implements the proposed automatic semantic bridging process;
 - Implements the proposed execution process;
 - Implements an operational graphical user interface for all previous modules.

Finally, it is relevant to return to the requirements initially defined in 2.5.7. For each of the requirements a short description is given concerning the support provided by the research of this thesis:

1. Identification, specification and representation of syntactic, schematic and semantic relations between distinct information semantics. This requirement is extensively supported by the work executed in this thesis, and in particular:
 - It is supported by the Semantic Bridging phase introduced in MAFRA;
 - The SBO permits the specification and representation of syntactic, schematic and semantic

relations. While, it is conceptually impossible to determine the degree of support provided by SBO, it is perceptible by third-part opinions that SBO features are relevant and sufficient for a large number of application and scenarios;

- The identification of the relations is supported by the automatic semantic bridging process presented in 7.3. Above all, the proposed process aims to reduce the number of generated (identified) SemanticBridges, but it is our perception that the solution is not sufficiently skilled and should be improved (Chapter 10).
2. Transformation of information exchanged among intervenients according to the specified syntactic, model and semantic relations. This requirement is referred in two research elements:
 - In the execution phase of MAFRA;
 - In the execution process described in Chapter 6, which fully supports the conceptualization made by SBO. Therefore, as observed for SBO, the developed transformation process supports and satisfy a large number of ontology mapping scenarios;
 3. Negotiation capabilities to reach consensus is supported by the MAFRA Cooperative Consensus Building module. However, no consistent research has been done so far, though it has been referred in 7.2 as it might benefice from the adoption of the multi-dimensional service-oriented architecture;
 4. Maintenance of the syntactic, schematic and semantic relations is supported by the MAFRA Evolution module, but no further research has been done in this subject;
 5. Integrate but minimize the human-being intervention in the mapping process, which suggests the adoption of a semi-automatic, human-supervised ontology mapping system. This requirement is only partially supported by the performed research. In particular:
 - The Domain Knowledge and Constraints module of MAFRA, which represents all automatic sources of knowledge and expertise that can be useful for the automation of the overall process;
 - In particular, the adopted notion of Multi-dimensional Service and Matcher are representations and embody relevant knowledge and expertise about respectively (i) the use of the transformation capabilities of the Service and (ii) the capability to determine a specific similarity between ontologies entities;
 - The automatic semantic bridging process is however the most relevant research subject concerning the minimization of the human-being participation in the process, while accepting and promoting the ultimate decision of the human-being;
 - The declarative, simple and compact conceptualization of SBO aims to reduce the human-being efforts and participation in the specification and representation process of the semantic relations;

- MAFRA Toolkit GUI, provides an interaction mechanism between the automated support and the human-being. Yet, at same time, by providing a simple and intuitive interface, it reduces the human-being efforts in the process.
6. Semantic web awareness. This requirement is widely referred during the thesis:
- SBO has been represented in RDFS and DAML+OIL, two of the most important representation mechanisms of the Semantic Web;
 - Ontology mapping document is represented in RDF (the basic representation model of the Semantic Web);
 - Ontologies are lifted to and manipulated in a RDFS-like representation language;
 - Source knowledge base is lifted and manipulated in a RDF representation language;
 - The Multi-Dimensional Service-Oriented Architecture, especially respecting the distributed, independent and pluggable modeling approach of Services. Services are developed, plugged and involve independently as required by the ontology mapping scenarios.

While the degree of provided support is difficult to determine, on the other hand, all the requirements have been addressed and most of them are at least partially supported by the work developed during this thesis.

9.3 Final remarks

Several evidences motivate the perception that a relevant, valid and useful research work has been done in the scope of this thesis:

- The large number of referred international conferences in which publications derived from this work have been accepted [Maedche *et al.*, 2002b; Maedche *et al.*, 2002a; Silva *et al.*, 2003; Silva & Rocha, 2002; Silva & Rocha, 2003a; Silva & Rocha, 2003b; Silva & Rocha, 2003c; Silva & Rocha, 2003d; Silva & Rocha, 2003e; Silva & Rocha, 2004a; Silva & Rocha, 2004b];
- The large number of research publications citing MAFRA, SBO, the execution process and the MAFRA Toolkit from very different research fields [Bruijn & Polleres, 2004; Ding *et al.*, 2003; Dou *et al.*, 2002; Janowicz & Riedman, 2004; Laleci *et al.*, 2004; Sinir *et al.*, 2004];
- The good opinions arising from the large number of application (and) experiences achieved with SBO and the MAFRA Toolkit by third-part research groups and projects;
- The constructive suggestions [Harmo-TEN] to improve research results and the MAFRA Toolkit.

In addition to the immediate results achieved, the work executed in the scope of this thesis has profound impact in the way research is now understood and practiced by the author. In particular, this thesis provided:

- Improved capabilities in the analysis and systematization of the research problems;
- Improved capabilities concerning the application of research methodology;
- Improved capabilities to engage and participate in research projects and discussions;
- Improved capability to report research work to the research community.

Chapter 10

ONGOING AND FUTURE RESEARCH

Based on the observed limitations raised by experiences and according to the research topics described during the thesis, this chapter describes current and future research directions in the ontology engineering domain in general and in ontology mapping in particular.

The following subjects are addressed in this chapter:

- Combination of Services;
- Abstraction of the extensional specification;
- Automatic semantic bridging process;
- Integration with/in other systems;
- Graphical user interface;
- Evolution;
- Negotiation;
- Development and code re-engineering;

- Standardization;
- Experiences and case tests.

Next sections address each of these topics.

10.1 Combination of Services

While the originally develop Services are sufficient for the majority of the experiences performed so far, it has been necessary to develop or refine some specific Services. While this is a foreseen idea, it has been noticed that, in some circumstances, the new Services correspond to the combination of the originals ones.

Combination of Services in the same SemanticBridge permits that a subset of the SemanticBridge parameters are used to calculate intermediary set of instances that will be applied in another parameter of the SemanticBridge. From this observation, a solution has been envisaged that involves a mechanism in which the instances of a particular Service/SemanticBridge parameter are provided as the outcome of a PropertyBridge. This conceptual solution seems to fulfill the requirements, yet does not seem very difficult to implement. It is evident that some modifications should occur:

- In SBO, at least a new type of relation between SemanticBridges should be specified, permitting the association of PropertyBridge with a specific SemanticBridge parameter. Adopting a generic parameter-based approach, it permits that even the non-Service-defined parameters (e.g. Conditions or the Extensional Specification elements) make use of this mechanism;
- In the Execution Engine, every SemanticBridge has a dependency order and an execution order. The dependency order concerns with the number of SemanticBridges that depend on it. The execution order defines the order based on which the SemanticBridge is executed in relation to the others. This situation is graphically represented in Figure 10.1: because PropertyBridge3 provides the input for PropertyBridge2, it should be executed before. In the same sense PropertyBridge3 is executed before SemanticBridge1.

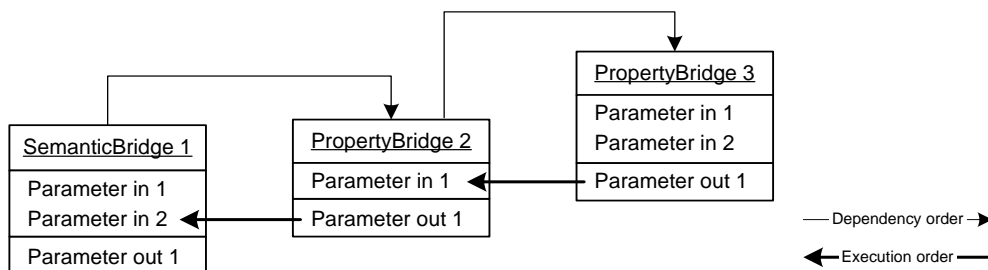


Figure 10.1 – Execution order between SemanticBridges

The dependency and execution order of the SemanticBridges represented in Figure 10.1 is shown in Table 10.1:

Table 10.1 – Dependency and execution orders between SemanticBridges

SemanticBridge	Dependency order	Execution order
SemanticBridge 1	0	3
PropertyBridge 2	1	2
PropertyBridge 3	2	1

Notice however that at least three constraints must hold:

- No cyclic dependencies should exist between any set of SemanticBridges, or a deadlock will occur at the execution phase;
- The number of output cardinality of the dependable PropertyBridge should conform to the number of the input cardinality of the dependent SemanticBridge. The possibility to use other PropertyBridges than 1:1 or n:1 cardinality PropertyBridges depends on the capabilities of the system concerning the association of output arguments to input arguments;
- Because the same PropertyBridge may be applied in multiple SemanticBridges, the same PropertyBridge may have multiple dependency and execution orders. In that sense, the PropertyBridge should be executed only once, and in the lowest execution order observed for every dependency.

In many aspects, the proposed approach corresponds to transform SBO into a functional language:

- It permits to reduce the development of new Services. Even if the Services development procedure is rather simple for a programmer, it is nevertheless time-consuming and requires some knowledge about the subject;
- It permits to improve immediate semantic relation capabilities because it may be possible to combine competencies from Services instead of developing new ones;
- It potentially reduces the readability and clarity of SemanticBridges in the sense that more elements are included in the SemanticBridge;
- The automation of the semantic bridging phase is harder to achieve in the sense that multiple Services will be associated with the same SemanticBridge.

Summarizing, it is important that the original declarative, simple and compact approach provided by SBO does not turn into a programmatic language.

10.2 Abstraction of the extensional specification elements

It is often necessary to define the same extensional specification element in different SemanticBridges. The most typical situation requires defining twice the same extensional

specification, which is not practical nor contributes for the readability, clarity and correction of the mapping document. In fact, because extensional specification is not an object but a set of `ConditionExpressions` in a specific context, the same set of `ConditionExpressions` are defined in multiple contexts. It often occurs to update the extensional specification in one `SemanticBridges` but not in the other, motivating a semantic mistake or at least a semantic incoherence, which will provoke execution errors. Therefore, it has been noticed the need to provide more intuitive representation and manipulation mechanisms of the extensional specification elements.

The envisaged solution suggests the adoption of the so called virtual entity. A virtual entity is a new type of entity conceptualized through the SBO and instantiated in the scope of the ontology mapping document. Adopting this approach, ontologies maintain their original content and the information about the ontology mapping is stored where it is required, i.e. in the ontology mapping document.

Virtual entity is defined according to the aggregation and combination of distinct ontology-defined entities. In particular, it is envisaged the specification of the three following kinds:

- Virtual Concept, that broadly corresponds to the idea of class in DL. Virtual Concept is defined by the combination of a source ontology concept and a set of extensional specifications constraints. Typically, a virtual concept would be defined in situations where an extensional specification would be necessary, i.e. when a source ontology concept is more generic than the most similar concept in the target ontology;
- Virtual Relation is the combination of relations that relate an ontology-defined concept and a virtual concept, or two virtual concepts. The definition of a virtual relation will occur in scenarios where the extensional specification is used in `PropertyBridges`, i.e. when a source ontology concept is related to an extensionally defined concept;
- Virtual Attribute is the combination of a set of ontology source attributes and a set of extensional specification elements. The definition of a virtual attribute will occur when it is necessary to access an attribute that occurs in the scope of a Virtual Concept.

During the semantic bridging phase, these entities will be treated as any ontology-defined concept, but at execution process it is necessary to perform the extensional specification query and filtering processes as referred in 6.3.

Because the instantiation of these entities is made in the scope of the ontology mapping document, it is necessary to enhance the GUI so these new SBO entities can be represented and manipulated. While this implementation should not be difficult, it should be combined with the new characteristics of the GUI (presented at Section 10.4).

10.3 Automatic semantic bridging process

Three main research directions are particularly envisaged in the near future concerning the improvement of the automatic semantic bridging process/system:

- Improvement of capabilities of Services to choose and decide the SemanticBridges. This may be achieved by refining the Services constraints, both manually and automatically. The automatic customization of Services constraints is envisaged as a major research topic in the near future. For that, the exploitation of machine learning techniques is envisaged. In particular, the approach would provide Services with the capabilities to observe/analyze the user decisions about its own proposed clusters and SemanticBridges. According to the changes performed by the user, the Service would be capable to automatically propose the update of its own conditions such an eventual re-execution of the process would result in the ontology mapping document accepted by the user. Yet, this is not an easy task, both because of the intrinsically ambiguous nature of the problem and because of the limited elements the Service can reason upon and customize. As a consequence, the Services automatic updates would easily run into contradiction, which should be avoided;

- Inclusion of new matchers into the system, either by adopting already existent matchers or by developing new ones. Many of the existent matchers ground on statistical and linguistic knowledge bases. These intrinsically ambiguous approaches tend to forward ambiguity to other phases of the process, in particular to clustering and bridging phases.

The research and development of new matchers capable to analyze and exploit the information resulting from well-specified, formal ontology engineering processes is envisaged as a good starting point. In particular, processes such as (formal) development, assembling, merging, evolution and mapping of ontologies provide either implicit or explicit information that can be useful in the automatic semantic bridging process. New matchers would be responsible for capturing that information into useful and meaningful means to be used by the Services;

- Specification and execution of a battery of tests and their comparison with the user-defined document and with other similar systems. This subject is address further in Section 10.7.

This is an on-going research topic, in which the capabilities and information provided by the FONTE assembling process [Harmo-TEN; Santos & Staab, 2003] is exploited by a specific matcher [Harmo-TEN; Silva *et al.*, 2004] into the clustering and semantic bridging processes.

10.4 Graphical user interface

The GUI component is currently the most continuously demanding and evolving part of the MAFRA Toolkit. This development especially occurs in the manipulation capabilities of graphical entities.

However, despite current ordinary limitations, other requirements will motivate strong development efforts of the GUI in the near future. In particular:

- Combination of Services, in case the approach described in 10.1 is adopted;
- Representation and manipulation of the Virtual Concept, Virtual Relation and Virtual Attribute as described in 10.2;
- GUI improvement so it is possible to automatically and with assisted, constrain the entities shown in the graph. This is a strong requirement arising in large ontology mapping scenarios.

For the moment the most realistic and promising approach is the seminal work of Stuckenschmidt and Klein on the field of ontology partition [Stuckenschmidt & Klein, 2004]. Adopting this approach, it would be necessary to research on the ontology partition process, especially concerning the assisted process of driving the partition according to:

- The content of the other ontology;
- The context defined by the (current) ontology mapping process.

Despite these envisaged improvements of the GUI, the approach must support the user-based semantic bridging phase and should benefit from the work done so far in the automatic semantic bridging process.

Yet, it starts to be perceived that the ontology engineering and in particular ontology mapping graphical user interface requirements form a very special case of research. In that sense, researching and developing on ontology engineering graphical user interfaces should be the focus of a specific research work.

Currently however, it is running an effort to combine into the same GUI the ontology assembling process developed in FONTE [Santos & Staab, 2003] and the ontology mapping process described in this thesis.

10.5 Evolution

It is not difficult for ontology mapping to become incoherent when a number of changes occur in the mapped ontologies. The evolution of the ontology mapping document may and should profit from previous ontology mapping documents and eventually from other information acquired and stored during the semantic bridging process. Evolution process is therefore substantially different from semantic bridging process, requiring further research and development.

Besides the literature on ontology evolution [Maedche *et al.*, 2003; Stojanovic *et al.*, 2002a] and versioning of ontologies [Klein *et al.*, 2002] a very important starting point should be the multi-dimensional service-oriented architecture, in the sense that, once again, the competence and know-

how to deal with specific changes are forwarded to Services. No research work in this topic is known or has been done in the scope of this thesis.

10.6 Common Consensus Building

As referred in 4.4.2, only residual research work exists on this topic, which turns this research very important but also more effort demanding.

Yet, while existent literature should be exploited, the initial research on this topic might focus on the potentialities of the multi-dimensional service-oriented architecture, as suggested for evolution and automatic semantic bridging processes.

Communication infrastructure and negotiation protocol are currently being developed in the scope of MAFRA Toolkit, but more advanced solutions are expected, especially in the field of meaning negotiation and machine learning, as previously suggested for the automatic semantic bridging process.

10.7 Integration with/in other systems

The research ideas proposed in this thesis are mostly implemented in the MAFRA Toolkit. While its application has succeeded in a variety of scenarios, its stand alone nature is not suited for the application in on-line interoperability scenarios. The problems arise especially due to the fact that in some ontology mapping scenarios [Laleci *et al.*, 2004; Sinir *et al.*, 2004], semantic bridging and execution phases are executed in completely distinct contexts. In these scenarios, the semantic bridging phase is typically an off-line process in which a set of domain experts establish the ontology mapping documents between two or more ontologies. The execution phase, on the contrary, is an on-line process in which several (running) entities require the transformation of the contents of their messages on the fly. In many scenarios, the off-line execution process is not a feasible solution because entities repositories are too large and change continuously.

In order to answer this requirement, two distinct, non-contradictory conceptual approaches are envisaged:

- Research on the integration of the proposed ideas into on-line systems/processes. Notice that this has been set as one the initial research topics of this thesis (Chapter 1), but due to the lack of ontology mapping research and tools it has been set apart in early stages of work.

However, because new technological paradigms surged meantime, new different approaches and solutions from those initially motivating this work are necessary. In fact, the Semantic Web and web services open new perspectives on the interoperability between different kinds of entities, compelling new approaches and solutions. Still, as stated in [Benjamins *et al.*, 2003], agent-based

systems and web services are neither contradictory nor incompatibles. Actually, both paradigms are complementary in many aspects and mutually profit from their combination.

In that sense, the conceptual research goal is to research and develop an intelligent and evolving interoperability facilitator system, by the combination of the ontology mapping approaches proposed in this thesis with the Semantic Web, web services and agent-based systems;

- Provide the implementation of the research ideas as reusable software packages instead of a stand alone application.

Some, more pragmatic ideas on how to provide a short-term prototype of the just described conceptual solutions are further addressed in 10.8.

10.8 Development and code re-engineering

Most of the research ideas proposed in this thesis have been implemented in MAFRA Toolkit, which despite the large number of scenarios and experiences it has been applied in, it is not a final product. On the contrary, it is constantly evolving.

As referred in 10.7, one of the limitations of MAFRA Toolkit is its stand alone, off-line nature. Pragmatically, in order to overcome this limitation, it is necessary to:

- Develop the functionalities such only parts of the repositories (messages content) are transformed;
- Develop the functionalities such the execution process can be called arbitrarily in time.

Two main development approaches may be followed, though both are compatible and advisable:

- Provide MAFRA Toolkit functionalities through a wrapping mechanism. In particular this would be provided by:
 - A web service that wraps the MAFRA Toolkit functionalities into an on-line application;
 - Generic web services interfaces for entities using MAFRA Toolkit through the web service;
- Develop and re-engineer MAFRA Toolkit code, such the envisaged functionalities are provided through a well-defined, stable and fully functional Application Program Interface (API). MAFRA Toolkit functionalities would then be provided as software packages (libraries) whose functionalities would be applied by software engineers as required by the application scenario.

However, these improvements are just a few of many other necessary or suggested by third-part research teams. In order to continuously support and improve the research work more generic development policies are being set with some of the third-part research teams.

These policies, derived from the development experiences had so far, distinguish two main areas of development:

- Incorporation of new functionalities. These new functionalities correspond to the implementation of the ideas resulting from the research efforts namely from those proposed in previous sections;
- Improving software packages, which corresponds to two main efforts:
 - Providing reliable, error-prone, stable and better performing solutions;
 - Providing implementations such they better conform to the requirements found in specific application scenarios.

While there is no current task assignment between development team, there are some on-going development efforts. In particular:

- MAFRA Toolkit code is currently being analyzed and systematized in order to be re-engineered into a robust and highly performing tool, by the E-Commerce Competence Center (EC3) research group (Vienna, Austria), in the scope of the Harmo-TEN project;
- MAFRA Toolkit is currently being re-engineered into a set of distinct software packages by Anna V. Zhdanova from DERI, Innsbruck, Austria;
- New developments in the automatic semantic bridging and negotiation processes are being currently pursued in the scope of SANSKI [SANSKI] and OntoMapper [OntoMapper] projects, by the GECAD team, Porto, Portugal.

10.9 Standardization

In early phases of SBO specification there was no concern in creating a representation language of ontology mapping documents that could be widely used or standardized.

However, currently, there are some efforts attempting to define an ontology mapping language for the Semantic Web, which are of great interest both personally and institutionally. In fact, considering the relevance of SBO in literature, it is perceived that it may serve as a good starting point, as referred in [Bruijn & Polleres, 2004].

In this sense, research contacts should be set with relevant institutions in this subject, such the ideas preconized in SBO and in this thesis are further applied in larger contexts.

10.10 Experiences and case tests

Some preliminary efforts have been carried out in the later stages of this thesis concerning the evaluation and comparison of ontology mapping systems and in special concerning SBO semantic expressivity and execution process performance. However, as referred through the last chapters, no formal experiences or comparison have been made so far, constraining a more factual and definite analysis.

In order to overcome this limitation, a strategy has been delineated based on three complementary tasks:

Establish a set of ontology mapping scenarios and KB to be used in the evaluation;

- Establish a set of experiences upon previous ontology mapping scenarios and KB;
- Establish a set of comparison functions between ontology mapping tools based on the results achieved in previous point.

In order to define and establish a wide consensus upon previous elements, some contacts have already been made with other research teams, namely with the teams.

Yet, more than any experience or comparison, the results should be published and disseminated as soon and widely as possible in order to promote the evaluation and discussion upon the subject.

10.11 Outlook

The ontology mapping process systematization provided by MAFRA distinguishes and suggests very important research areas in which considerable efforts should be putted in order to turn ontology mapping systems into a useful reality. This section systematizes this research areas while enumerating some others derived from the experiences developed with MAFRA Toolkit and from the feedback received from third-part research teams.

Besides the achieved results presented in Chapter 9 and the research literature cited during the thesis, research work on ontology mapping domain and in the context of the Semantic Web is still in its early phases. In fact, while very important work has been done in recent years, the area of standardization of technology attained the largest part, while neglecting other very important areas.

It is perceptible that the ontology mapping problem is know acquiring the necessary relevance by many research groups, which suggests that relevant results will be achieved in the near future.

FOURTH PART

Annex 1

RELATIONAL DATA MODEL

This annex describes the relational data model and the relational algebra. A brief comparison with the ontology data model is also provided. According to the perceived differences a method will be described concerning the transformation of ontologies and corresponding knowledge bases (as formalized in 3.3) into relational schemas and relations, respectively. Once knowledge bases can be treated as relations, the relational algebra will be described, providing a formal mechanism to query and access knowledge bases.

This annex does not intend to describe or analyze the relational data model, but to present a perspective that allows the understanding of the approach developed during Chapter 6. For finer descriptions of the relational data model please refer to [Bruijn & Polleres, 2004; Codd, 1970; Date, 2003], for example.

A 1.1 Building blocks

Relational data model is based on the notion of n-ary mathematical relation $R(x_1, x_2, \dots, x_n)$, which is equivalent to $(x_1, x_2, \dots, x_n) \in R$. Formally, an n-ary relation is defined by $R(X_1, X_2, \dots, X_n, G(R))$, where X_1, X_2 and X_n are sets, and $G(R)$ is the graph of R . $G(R)$ is a subset of the Cartesian product of X_1, X_2 and X_n , which represents the actual set of values in R . Mathematical background and formalisms have been firstly adopted and applied by Codd in 1970 [Codd, 1970] in the specification of the relational data model, which served the basis for further formalization of data models, including those that existed prior to the relation data model (e.g. hierarchical and network data models).

The main concept in the relational data model is the therefore the relation, corresponding to its homonym mathematical concept, and to the concept of ontology. Informally, a relation is a table constituted by:

- A header row of attribute names, which corresponds in mathematical notation to the names of the sets of the relation, and in ontology notation to properties.
- Others rows in the table, called tuples, which corresponds in mathematical notation to elements in $G(R)$, and in the knowledge base notation to instances.

Example A 1.1 - Generic transformation of a knowledge base into relations

Consider the following knowledge base (also applied in Chapter 6).

$$\begin{aligned} \mathcal{I}_{O1} &= \{i_{1.1}, i_{1.2}, i_{1.3}, i_{1.4}, i_{1.5}, f_{1.1}, f_{1.2}, f_{1.3}, e_{1.1}, e_{1.2}, e_{1.3}, e_{1.4}\} \\ \text{inst}\mathcal{C}_{O1} &= \left\{ \begin{array}{l} \text{Individual}(i_{1.1}), \text{Individual}(i_{1.2}), \text{Individual}(i_{1.3}), \text{Individual}(i_{1.4}), \\ \text{Individual}(i_{1.5}), \text{Family}(f_{1.1}), \text{Family}(f_{1.2}), \text{Family}(f_{1.3}), \\ \text{Event}(e_{1.1}), \text{Event}(e_{1.2}), \text{Event}(e_{1.3}), \text{Event}(e_{1.4}), \text{Event}(e_{1.5}) \end{array} \right\} \\ \text{inst}\mathcal{P}_{O1} &= \left\{ \begin{array}{l} \text{gender}(i_{1.1}, "M"), \text{gender}(i_{1.2}, "F"), \text{gender}(i_{1.3}, "F"), \text{gender}(i_{1.4}, "M"), \\ \text{gender}(i_{1.5}, "F"), \text{birth}(i_{1.1}, e_{1.5}), \text{date}(e_{1.5}, 1769), \\ \text{name}(i_{1.1}, "Napoleon Bonapart"), \text{name}(i_{1.2}, "Joséphine de Tasher"), \\ \text{name}(i_{1.3}, "Marie - Louise de Austria"), \text{name}(i_{1.4}, "William Clinton"), \\ \text{name}(i_{1.5}, "Hillary Rodham"), \\ \text{spouseIn}(i_{1.1}, f_{1.1}), \text{spouseIn}(i_{1.1}, f_{1.2}), \text{spouseIn}(i_{1.2}, f_{1.1}), \\ \text{spouseIn}(i_{1.3}, f_{1.2}), \text{spouseIn}(i_{1.4}, f_{1.3}), \text{spouseIn}(i_{1.5}, f_{1.3}), \\ \text{marriage}(f_{1.1}, e_{1.1}), \text{divorce}(f_{1.1}, e_{1.2}), \\ \text{marriage}(f_{1.2}, e_{1.3}), \text{marriage}(f_{1.3}, e_{1.4}), \\ \text{date}(e_{1.1}, 1796), \text{date}(e_{1.2}, 1810), \text{date}(e_{1.3}, 1810), \text{date}(e_{1.4}, 1975) \end{array} \right\} \end{aligned}$$

Table A 1.1 represents the Individual relation, whose attributes are ID, name, gender and spouseIn.

Table A 1.1 - The Individual relation (concept) and some tuples (instances)

ID	name	gender	spouseIn
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.1}$
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.2}$
$i_{1.2}$	“Joséphine de Tasher”	“F”	$f_{1.1}$
$i_{1.3}$	“Marie-Louise de Austria”	“F”	$f_{1.2}$
$i_{1.4}$	“William Clinton”	“M”	$f_{1.3}$
$i_{1.5}$	“Hillary Rodham”	“F”	$f_{1.3}$

More formally, relational data model comprehends nine fundamental concepts:

1. Domain, is set of atomic (indivisible) values, corresponding to the X_1 , X_2 and X_n mentioned in the mathematical description;
2. Attribute, denoted by a_i , corresponds to the property element of the ontology definition introduced in 3.3. Attributes identify and characterize columns of the relation. Each attribute a_i , has a domain: $Dom(a_i)$;
3. Relation schema, denoted by $R(a_1, a_2, \dots, a_n)$, where R is the name of the relation, and a_i , a_2 and a_n are the attributes of the relation. Considering relation presented in Table A 1.1, relation schema would be *Individual*($ID, name, gender, spouseIn$);
4. Degree of a relation is the number of the attributes in a relation. For example, the degree of the Individual relation presented in Table A 1.1 is 4;
5. Relational Database Schema, denoted by $S = \{R_1, R_2, \dots, R_n\}$ is a set of relation schemas of the same database;
6. Tuple of a relation, denoted by $t := (v_1, v_2, \dots, v_n)$, is an ordered set of values such $v_i \in Dom(a_i)$. For example, $t_1 = (i_{1.1}, "Napoleon Bonapart", "M", f_{1.1})$ is a tuple of the Individual relation presented in Table A 1.1;
7. Relation instance is the set of tuples that conform to a certain relation schema, and corresponds to $G(R)$ in the mathematical notation. From the relation presented in Table A 1.1, relation instance is the set composed by the tuples representing each row of the table, except the header row;
8. Primary key is the set of attributes of a relation schema that once instantiated are unique in a relation and therefore univocally identify the relation tuple in the table. In the case of relation presented in Table A 1.1, the primary key is either $\{ID\}$ or $\{ID, spouseIn\}$;
9. Foreign key is the set of attributes in certain relation that corresponds to the primary key of another relation schema.

Due to its mathematical background, relations are representable as set of tuples. Moreover, as mathematical relations, database relations are often represented only by the graph of the relation (i.e. $G(R)$). For example, the relation presented in Table A 1.1 would be represented in set nomenclature as:

$$R_1 = \left\{ \begin{array}{l} (i_{1.1}, "Napoleon Bonapart", "M"), \\ (i_{1.2}, "Joséphine de Tasher", "F"), (i_{1.3}, "Marie - Louise de Austria", "F"), \\ (i_{1.4}, "William Clinton", "M"), (i_{1.5}, "Hillary Rodham", "F") \end{array} \right\}$$

A 1.2 Relational data model Vs. Ontology data model

This section describes the method used to manage schemas defined using the ontology data model presented in 3.3, as schemas defined using the relational data model. For finer details on the relations between both models please refer to literature in the area (e.g. [Motik *et al.*, 2003; Stojanovic *et al.*, 2002b]).

Despite the terminological differences previously enumerated, the ontology data model and relational data model are not immediately equivalent or compatible.

According to the description of relational model made in section A 1.1 and to the ontology formalization presented in 3.3, a comparison of terminology between both data models results in Table A 1.2:

Table A 1.2 – Terminological comparison between relational and ontology data models

Ontology data model	Relational data model
Ontology	Relational database schema
Concept, its Properties and their Instances	Relation
Concept and its Properties	Relation schema
Knowledge base	Union of all relations of a database
Concept	Relation name
Property	Attribute name
Domain	Name of the relation schema of the attribute
Range	Domain of an attribute
Concept instance	Tuple of a relation
Property instance	Value of a tuple of a relation
Concept instances	Relation instance

Normally, relational databases are structured according to the third normal form (3NF)⁵⁹, while ontologies rarely conform to it. However, relations non-conformant to the 3NF are still valid relations. The same happens with ontologies.

A 1.3 Translating ontology and knowledge base into relational model

Due to these differences, in order to apply relational algebra to ontology models, three adaptations have been developed specially in the following processes:

- Translation of ontology entities into relation schema;
- Translating concept instances and its properties into relations;
- Representation of forward and backward Paths in the relation schema.

Each of these processes is described in next sections.

A 1.3.1 Translation of ontology entities into relation schema

A relation schema corresponds to a concept and to the properties whose domain is the concept. The concept corresponds to the relation name while its properties are translated into the attributes of the relation schema.

Relation schema attributes do not corresponds to the properties names only, but instead to the combination of *Domain/Predicate/Range*, corresponding to the definition of the SBO Step concept.

Moreover, notice that every concept instance is univocally identified by a name (e.g. $i_{1.1}$), commonly referred as identifier, which serves to inter-relate concept instances. This identifier not only represents the concept instance but also characterizes it. In that sense the identifier is translated into the relation schema as the ID property whose range is Literal (Table A 1.3).

A 1.3.2 Normalizing concept instances

A source concept instance corresponds to the combination of all values of its properties, i.e. the Cartesian product of all property values, including its identification.

Example A 1.2 – Normalization of concept instances

Considering the Individual instance $i_{1.1}$ of the knowledge base, it corresponds to the table-based representation of Table A 1.3:

⁵⁹ Despite normal form range from first normal form (1NF) to fifth normal form (5NF), database designers typically intent to structure databases conforming to the 3NF only. Third normal form is also known as Boyce-Codd Normal Form (BCNF).

Table A 1.3 - Table-based representation of $i_{1.1}$ source concept instance

Individual/ID/ Literal	Individual/name/ Literal	Individual/gender/ Literal	Individual/spouseIn/ Family
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.1}$
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.2}$

A relation corresponds to all concept instances and their properties values. To refer to that relation or concept instances, the name of the ontology concept is used (e.g. Family, or O1:Family).

Example A 1.3 – Table-based representation of concept instances

Considering previously presented KB, the instances of the Family concept and their properties are represented as Table A 1.4:

Table A 1.4 - Table-based representation of all Family instances (Family relation)

Family/ID/Literal	Family/marriage/Event	Family/divorce/Event
$f_{1.1}$	$e_{1.1}$	$e_{1.2}$
$f_{1.2}$	$e_{1.3}$	
$f_{1.3}$	$e_{1.4}$	

A 1.3.3 Representation of forward and backward Paths in the relation schema

A Path is considered backward if it is composed by at least one Step whose direction attribute is set to “backward”. In that sense, the intended adaptation concerns to support “backward Steps” and not backward Paths.

Like forward Steps are represented as attributes of the relation schema, backward Steps are also represented as attributes of the relation schema, but using the coherent notation: *Domain \ Predicate \ Range*. Because backward Steps are not a standard relational model issue, a specific manipulation mechanism has been researched and specified during this thesis. Deeper details on the manipulation of “backward Steps” can be found in 6.2.

A 1.4 Relational algebra

Relational algebra is the mathematical model that permits reasoning upon schemas respecting the relational data model. Once ontologies and respective knowledge bases can be grounded to relational data model, the relational algebra is very useful in formally manipulating them.

The relational algebra is the set of operations on relations (tables) that permits to manipulate relations (i.e. the relation schema and/or relation tuples). In both cases, the result is again a relation that can be applied in further operations.

The relation algebra comprehends six fundamental operations:

1. The Selection operation, denoted by $\sigma_c R$, is a unary operation that corresponds to filter the tuples of the relation according to the arbitrary Boolean expression c . The result of a selection does not affect the relation schema (concept) but the tuples (instances).

Example A 1.4 – Selection operation

Consider the relation *Individual* and their instances presented in Table A 1.1. The selection $\sigma_{\text{Individual} / \text{gender} / \text{Literal} = "F"} \text{Individual}$ ⁶⁰ results in Table A 1.5:

Table A 1.5 - Result of the $\sigma_{\text{Individual} / \text{gender} / \text{Literal} = "F"} \text{Individual}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family
i _{1.2}	“Joséphine de Tasher”	“F”	f _{1.1}
i _{1.3}	“Marie-Louise de Austria”	“F”	f _{1.2}
i _{1.5}	“Hillary Rodham”	“F”	f _{1.3}

2. The Projection operation, denoted by $\pi_{a_1, \dots, a_n} R$ is a n-unary operation that results in a new relation schema (concept) with the attributes a_1, \dots, a_n from relation R , and the corresponding tuples.

Example A 1.5 – Projection operation

The operation $\pi_{\text{name,gender}}$ Table A 1.1 results in the relation represented in Table A 1.6:

Table A 1.6 – Result of the $\pi_{\text{name,gender}}$ Table A 1.1 operation

Individual/name/Literal	Individual/gender/Literal
“Napoleon Bonapart”	“M”
“Napoleon Bonapart”	“M”
“Joséphine de Tasher”	“F”
“Marie-Louise de Austria”	“F”
“Hillary Rodham”	“F”
“William Clinton”	“M”

3. The Cartesian Product operation, denoted by $R \times S$, is a binary operation upon two relations (R and S), that corresponds to the homonym operation in the set theory. It combines all tuples in R with all tuples in S . The attributes from R and S must be disjoint (different names) or fully qualified names in the resulting relation will be used.

⁶⁰ In order to maintain examples as simple as possible, from now on attributes of relations are referred only by its name (e.g. $\sigma_{\text{gender} = "F"} \text{Individual}$) instead of the Domain/Predicate/Object form. However, in ambiguous situations the fully qualified nomenclature will be used.

Example A 1.6 – Cartesian Product operation

Consider the Individual relation presented in Table A 1.5 and the Family relation presented in Table A 1.7:

Table A 1.7 - Family relation

Family/ID/Literal	Family/marriage/Event
f _{1.1}	e _{1.1}
f _{1.2}	e _{1.2}
f _{1.3}	e _{1.4}

The operation Table A 1.5×Table A 1.7 will result in the new relation represented in Table A 1.8:

Table A 1.8 – Result of the Table A 1.1×Table A 1.7 operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family	Family/ ID/ Literal	Family/ marriage/ Event
i _{1.2}	“Joséphine de ...”	“F”	f _{1.1}	f _{1.1}	e _{1.1}
i _{1.2}	“Joséphine de ...”	“F”	f _{1.1}	f _{1.2}	e _{1.2}
i _{1.2}	“Joséphine de ...”	“F”	f _{1.1}	f _{1.3}	e _{1.4}
i _{1.3}	“Marie-Louise de ...”	“F”	f _{1.2}	f _{1.1}	e _{1.1}
i _{1.3}	“Marie-Louise de ...”	“F”	f _{1.2}	f _{1.2}	e _{1.2}
i _{1.3}	“Marie-Louise de ...”	“F”	f _{1.2}	f _{1.3}	e _{1.4}
i _{1.5}	“Hillary ...”	“F”	f _{1.3}	f _{1.1}	e _{1.1}
i _{1.5}	“Hillary ...”	“F”	f _{1.3}	f _{1.2}	e _{1.2}
i _{1.5}	“Hillary ...”	“F”	f _{1.3}	f _{1.3}	e _{1.4}

Semantically, the Cartesian product is normally meaningless, and therefore it is not usually used directly, but as combination with the Selection operation. This corresponds to the non-standard Join operation (refer to A 1.5).

- The Union operation, denoted by $R \cup S$, is a binary operation upon two relations (R and S), that corresponds to the union of tuples from both relations, which implies that both relations have the same attributes. In case repeated tuples are found, only one is copied into the result.

Example A 1.7 – Union operation

Consider the relation represented in Table A 1.1 and the relation presented in Table A 1.9:

Table A 1.9 - Another Individual relation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family
i _{1.1}	“Napoleon Bonapart”	“M”	f _{1.1}
i _{1.6}	“Andy Warhol”	“M”	

The operation $\text{Table A 1.1} \cup \text{Table A 1.9}$ results in Table A 1.10:

Table A 1.10 – Result of the $\text{Table A 1.1} \cup \text{Table A 1.9}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.1}$
$i_{1.1}$	“Napoleon Bonapart”	“M”	$f_{1.2}$
$i_{1.2}$	“Joséphine de Tasher”	“F”	$f_{1.1}$
$i_{1.3}$	“Marie-Louise de Austria”	“F”	$f_{1.2}$
$i_{1.4}$	“William Clinton”	“M”	$f_{1.3}$
$i_{1.5}$	“Hillary Rodham”	“F”	$f_{1.3}$
$i_{1.6}$	“Andy Warhol”	“M”	

5. The Difference operation, denoted by $R - S$, is a binary operation that calculates the tuples of R that are not present in S .

Example A 1.8 – Difference operation

Consider Table A 1.1 and Table A 1.9 upon the Individual relation. The operation $\text{Table A 1.9} - \text{Table A 1.1}$ would result in Table A 1.11:

Table A 1.11 – Result of the $\text{Table A 1.9} - \text{Table A 1.1}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family
$i_{1.6}$	“Andy Warhol”	“M”	

6. The Rename operation, denoted by $\rho_{a/b}R$, is a unary operation upon the schema of the relation and not upon the instances. The result is a relation with the same set of tuples of R whose attribute b is renamed to a .

Example A 1.9 – Rename operation

Consider the operation $\rho_{sex/gender} \text{Individual}$ upon Table A 1.11. The result would be Table A 1.12:

Table A 1.12 – Result of the $\rho_{sex/gender} \text{Table A 1.11}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ sex/ Literal	Individual/ spouseIn/ Family
$i_{1.6}$	“Andy Warhol”	“M”	

Other operations are commonly recognized in relational model, even if they can be achieved through the combination of the six previous operations.

A 1.5 Complementary operations

Three complementary operations are considered useful for the work described in this thesis: The Intersection operation, denoted by $R \cap S$, can be expressed by $R - (R - S)$, and represents the set of tuples that are present simultaneously in R and S .

Example A 1.10 – Intersection operation

Consider the relations of Table A 1.9 and Table A 1.11, and the operation Table A 1.9 \cap Table A 1.11. The result would be the relation in Table A 1.13:

Table A 1.13 – Result of the Table A 1.9 \cap Table A 1.11

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family
i _{1.6}	“Andy Warhol”	“M”	

- The Theta Join operation, denoted by $R \theta_c S$ can be expressed by $\sigma_c (R \times S)$, and represents the Cartesian product of the tuples for which condition c holds true.

Example A 1.11 – Theta Join operation

Consider the Individual relation presented in Table A 1.5 and the Family relation presented in Table A 1.7. Because the attribute Individual/spouseIn/Literal corresponds to the attribute ID of the Family relation, it is semantically correct to join both tables when these two attributes are equivalent. The result of operation Table A 1.5 $\theta_{spouseIn=ID}$ Table A 1.7 is presented in Table A 1.14:

Table A 1.14 – Result of the Table A 1.5 $\theta_{spouseIn=ID}$ Table A 1.7 operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Family	Family/ ID/ Literal	Family/ marriage/ Literal
i _{1.2}	“Joséphine de ...”	“F”	f _{1.1}	f _{1.1}	e _{1.1}
i _{1.3}	“Marie-Louise de ...”	“F”	f _{1.2}	f _{1.2}	e _{1.3}
i _{1.5}	“Hillary ...”	“F”	f _{1.3}	f _{1.3}	e _{1.4}

- The Natural Join operation, denoted by $R^*_{(R \text{ join attributes}),(S \text{ join attributes})} S$, is similar to Theta Join except that the join condition is based on equality between columns values of both R and S expressed in join attributes.

Besides the condition issue, another particularity of the natural join in comparison with theta join is the fact that the join attributes of second relation are excluded from the resulting relation, because they are redundant. This is admissible because both columns have the same values.

Example A 1.12 – Natural Join operation

The result of operation $\text{Table A 1.10} \overset{*}{|}_{(spouseIn),(ID)} \text{Table A 1.7}$ is the relation presented in Table A 1.15:

Table A 1.15 – Result of the $\text{Table A 1.10} \overset{*}{|}_{(spouseIn),(ID)} \text{Table A 1.7}$

Individual/ ID/ Literal	Individual/name/Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Literal	Family/ marriage/ Event
i _{1.1}	“Napoleon Bonapart”	“M”	f _{1.1}	e _{1.1}
i _{1.1}	“Napoleon Bonapart”	“M”	f _{1.2}	e _{1.3}
i _{1.2}	“Joséphine de Tasher”	“F”	f _{1.1}	e _{1.1}
i _{1.3}	“Marie-Louise de Austria”	“F”	f _{1.2}	e _{1.2}
i _{1.4}	“William Clinton”	“M”	f _{1.3}	e _{1.3}
i _{1.5}	“Hillary Rodham”	“F”	f _{1.3}	e _{1.3}

4. The Left and Right Join operations, denoted by $R |_{*c} S$ and $R \overset{*}{|}_c S$ respectively, are similar to the natural join presented previously except that the tuples whose join columns do not match are kept in resulting relation. If left join is used, all tuples from R are maintained even if no relation exist between the tuple from R and tuples from S . If right join is used, all tuples from S are maintained even if no relation exists between the tuple form S and tuples from R .

Example A 1.13 – Left Join operation

Considering previous example, if the left join operation $\text{Table A 1.10} |_{*c} \text{Table A 1.7}$ is used instead of natural join operation, the result would be the relation presented in Table A 1.16

Table A 1.16 – Result of the $\text{Table A 1.10} |_{*c} \text{Table A 1.7}$ operation

Individual/ ID/ Literal	Individual/ name/ Literal	Individual/ gender/ Literal	Individual/ spouseIn/ Literal	Family/ ID/ Literal	Family/ marriage/ Event
i _{1.1}	“Napoleon ...”	“M”	f _{1.1}	f _{1.1}	e _{1.1}
i _{1.1}	“Napoleon ...”	“M”	f _{1.2}	f _{1.2}	e _{1.3}
i _{1.2}	“Joséphine de ...”	“F”	f _{1.1}	f _{1.1}	e _{1.1}
i _{1.3}	“Marie-Louise de ...”	“F”	f _{1.2}	f _{1.2}	e _{1.2}
i _{1.4}	“William ...”	“M”	f _{1.3}	f _{1.3}	e _{1.3}
i _{1.5}	“Hillary ...”	“F”	f _{1.3}	f _{1.3}	e _{1.3}
i _{1.6}	“Andy ...”	“M”			

Despite this operations are derived from the Natural Join presented above, in the scope of this work these operators do not exclude the redundant attribute (as illustrated in previous example) unless they are used between two relations whose relation schemas are exactly the same.

Other relational operations are commonly recognized and applied in nowadays DBMS, but the operations described in this annex are sufficient to demonstrate the approach and methods developed in context of this thesis.

A 1.6 Outlook

In this annex four main issues have been addressed:

- A light introduction to the relational data model, especially its building blocks and core principles;
- A comparison between the relational data model and the (adopted) ontology data model, including a method to transform ontology schemas into compliant relational schemas;
- Translation method between ontology and knowledge based entities and the relation model entities;
- A brief introduction to the relational algebra, that permits to formally manipulate relations and (from now on) knowledge bases.

BIBLIOGRAPHY

Arpírez, J. C.; Corcho, O.; Fernández López, M. and Gómez-Pérez, A. (2001); "WebODE: a scalable workbench for ontological engineering"; Proceedings of the International Conference on Knowledge Capture, 6-13; Victoria (BC), Canada.

Artemis; "Artemis: A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information"; <http://www.srdc.metu.edu.tr/webpage/projects/artemis>.

Atzeni, P. and Torlone, R. (1995); "Schema translation between heterogeneous data models in a lattice framework"; Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics, 345-364; Atlanta (GA), USA.

Bailin, S. C. and Truszkowski, W. (2001); "Ontology Negotiation between Agents Supporting Intelligent Information Management"; Proceedings of the Workshop on Ontologies in Agent Systems at the 5th International Conference on Autonomous Agents, 13-20; Montreal, Canada.

Bayardo, R. J.; Bohrer, W.; Brice, R.; Cichocki, A.; Fowler, A.; Helal, A.; Kashyap, V.; Ksiezyk, T.; Martin, G.; Nodine, M.; Rashid, M.; Rusinkiewicz, M.; Shea, R.; Unnikrishnan, C.; Unruh, A. and Woelk, D. (1997); "InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments"; Proceedings of the ACM SIGMOD International Conference on Management of Data, 195-206.

Bechhofer, S.; Horrocks, I.; Goble, C. and Stevens, R. (2001); "OILED: a reason-able ontology editor for the semantic web"; Proceedings of the Joint German/Austrian conference on Artificial Intelligence, 396-408; Vienna, Austria.

Beneventano, D.; Bergamaschi, S.; Fergnani, A.; Guerra, F.; Vincini, M. and Montanari, D. (2003); "A Peer-To-Peer Agent-Based Semantic Search Engine"; Proceedings of the 11th Italian Symposium on Advanced Database Systems (SEBD 2003), 367-378; Cetraro (CS), Italy.

Beneventano, D.; Bergamaschi, S.; Guerra, F. and Vincini, M. (2001); "The MOMIS Approach to Information Integration"; Proceedings of the Proceedings of the International Conference on Enterprise Information Systems, 194-198; Setúbal, Portugal.

- Benjamins, R.; Contreras, L. and Prieto, J. A. (2003); "Agents and the Semantic Web"; AgentLink News; 13 10-11; AgentLink.
- Benjamins, R.; Fensel, D. and Gómez-Pérez, A. (1998); "Knowledge Management through Ontologies"; Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98); Basel, Switzerland.
- Bergamaschi, S.; Castano, S. and Vincini, M. (1999); "Semantic Integration of Semistructured and Structured Data Sources"; SIGMOD Record; 28(1) 54-59; ACM Press.
- Berners-Lee, T. and Fischetti, M. (1999); "Weaving the Web The Original Design and Ultimate Destiny of the World Wide Web".
- Bernstein, P. A. and Rahm, E. (2001); "On Matching Schemas Automatically"; MSR-TR 2001-17; Microsoft Research.
- Broekstra, J.; Kampman, A. and van Harmelen, F. (2002); "Sesame: An Architecture for Storing and Querying RDF Data and Schema Information" Spinning the Semantic Web; 197-222; MIT Press.
- Bruijn, J. and Polleres, A. (2004); "Towards an Ontology Mapping Specification Language for the Semantic Web"; DERI TR 2004-06-30.
- Cattel, R. G.; Barry, D.; Berler, M.; Eastman, J.; Jordan, D.; Russell, C.; Schadow, O.; Stanienda, T. and Velez, F. (2000); "The Object Database Standard: ODMG 3.0"; Morgan Kaufmann.
- Chalupsky, H. (2000); "OntoMorph: A Translation System for Symbolic Knowledge"; Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, 471-482; Breckenridge (CO), USA.
- Codd, E. F. (1970); "A Relational Model of Data for Large Shared Data Banks"; Communications of the ACM; 13(6) 377-387; ACM.
- Coenen, F.; Eaglestone, B. and Ridley, M. (1999); "Validation, Verification, and Integrity in Knowledge and Database Systems: Future Directions" Validation and Verification of Knowledge-Based Systems: Theory, Tools and Practice; 297-312; Kluwer.
- Cohen, P. R. and Levesque, H. J. (1995); "Communicative Actions for Artificial Agents"; Proceedings of the First International Conference on Multi-Agent Systems, 65-72; San Francisco (CA), USA.
- Critchlow, T.; Ganesh Madhavan and Musick, R. (1998); "Automatic Generation of Warehouse Mediators Using an Ontology Engine"; Proceedings of the 5th Workshop on Knowledge Representation meets Databases, 8.1-8.8; Seattle (WA), USA.
- Crubézy, M. and Musen, M. A. (2003); "Ontologies in Support of Problem Solving" Handbook on Ontologies; 321-341; Springer-Verlag.
- Crubézy, M.; Pincus, Z. and Musen, M. A. (2003); "Mediating Knowledge between Application Components"; Proceedings of the Workshop on Semantic Integration of the International Semantic Web Conference; Sanibel Island (FL), USA.
- Date, C. J. (2003); "Introduction to Database Systems"; Addison-Wesley.
- Decker, S.; Erdmann, M.; Fensel, D. and Studer, R. (1999); "Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information"; Proceedings of the 8th Working Conference on Database Semantics, 351-369; Rotorua, New Zealand.
- Ding, Y.; Fensel, D.; Klein, M. and Omelayenko, B. (2002); "The semantic web: yet another hip?"; Data and Knowledge Engineering; 41(2-3) 205-227; Elsevier Science.
- Ding, Y.; Fensel, D.; Klein, M.; Omelayenko, B. and Schulten, E. (2003); "The role of ontologies in eCommerce" Handbook on Ontologies; Springer.
- Dionísio, N.; Marshall, I. and Safar, E. (2001); "Using Hyponym Branching Similarity Measures Comparable to Statistical Alternatives for Word Sense Disambiguation"; Proceedings of the Recent Advances in Natural Language Processing; Tzigov Chark, Bulgaria.

- Doan, A.; Domingos, P. and Halevy, A. (2001); "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach"; Proceedings of the ACM SIGMOD International Conference on Management of Data, 509-520; Santa Barbara (CA), USA.
- Doan, A.; Madhavan, J.; Domingos, P. and Halevy, A. (2002); "Learning to map ontologies on the Semantic Web"; Proceedings of the World-Wide Web Conference; Honolulu, Hawaii, USA.
- Dou, D.; McDermott, D. and Qi, P. (2003); "Ontology translation on the semantic web"; Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics, 952-969; Catania (Sicily), Italy.
- Dou, D.; McDermott, D. and Qi, P. (2002); "Ontology translation by ontology merging and automated reasoning"; Proceedings of the EKAW Workshop on Ontologies for Multi-Agent Systems, 3-18; Sigüenza, Spain.
- Fensel, D. (2001); "Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce"; Springer-Verlag.
- Fensel, D.; Benjamins, R.; Motta, E. and Wielinga, B. (1999); "UPML: A Framework for Knowledge system reuse"; Proceedings of the International Joint Conference on Artificial Intelligence, 16-23; Stockholm, Sweden.
- Fensel, D.; Horrocks, I.; van Harmelen, F.; Decker, S. and Klein, M. (2000); "OIL in a nutshell"; Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'2000), 1-16.
- Fensel, D.; van Harmelen, F.; Ding, Y.; Klein, M.; Akkermans, H.; Broekstra, J.; Kampman, A.; van der Meer, J.; Sure, Y.; Studer, R.; Krohn, U.; Davies, J.; Engels, R.; Iosif, V.; Kiryakov, A.; Lau, T.; Reimer, U. and Horrocks, I. (2003); "On-To-Knowledge: Semantic Web Enabled Knowledge Management".
- Fodor, O.; Dell'Erba, M.; Ricci, F.; Spada, A. and Werthner, H. (2002); "Conceptual Normalisation of XML Data for Interoperability in Tourism"; Proceedings of the Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002, 69-76.
- Fox, M. and Gruninger, M. (1997); "On Ontologies and Enterprise Modelling"; Proceedings of the International Conference on Enterprise Integration Modelling Technology; Torino, Italy.
- Gedcom; "Gedcom ontology"; <http://www.daml.org/2001/01/gedcom/gedcom.daml>.
- Gennari, J. H.; Tu, S. W.; Rothenfluh, T. E. and Musen, M. A. (1994); "Mapping Domains to Methods in Support of Reuse"; International Journal of Human-Computer Studies;(41) 399-424; Academic Press.
- Gentology; "Gentology ontology"; <http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml>.
- Global Exchange Services (2003); "Web Services White Paper"; Global eXchange Services.
- Goh, C. H. (1997); "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources"; PhD dissertation; MIT.
- Gruber, T. R. (1993b); "Towards Principles for the Design of Ontologies Used for Knowledge Sharing"; Proceedings of the Formal Ontology in Conceptual Analysis and Knowledge Representation, 907-928; Deventer, Netherlands.
- Gruber, T. R. (1993a); "A translation approach to portable ontology specifications"; Journal of Knowledge Acquisition; 5(2) 199-220; Academic Press.
- Gruninger, M. and Fox, M. (1995); "Methodology for the Design and Evaluation of Ontologies"; Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing in IJCAI-95; Montreal, Canada.
- Guarino, N. (1994); "The Ontological Level" Philosophy and the Cognitive Science; 443-456; Hölder-Pichler-Tempsky.

- Guarino, N. (1997a); "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration"; Proceedings of the International Summer School, 139-170; Frascati, Italy.
- Guarino, N. (1997b); "Understanding, Building and Using Ontologies: A Commentary to Using Explicit Ontologies in KBS Development by van Heijst, Schreiber, and Wielinga"; International Journal of Human-Computer Studies; 46(2/3) 293-310; Elsevier.
- Guarino, N. and Giaretta, P. (1995); "Ontologies and Knowledge Bases: Towards a Terminological Clarification" Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing; 25-32; IOS Press.
- Guarino, N. and Welty, C. (2000); "Towards a methodology for ontology-based model engineering"; Proceedings of the Workshop on Model Engineering at ECOOP-2000, 1-6; Cannes, France.
- Hagel, J. (2002); "The strategic value of Web services"; McKinsey.
- Halevy, A. (2001); "Answering queries using views: a survey"; The VLDB Journal The International Journal on Very Large Data Bases; 10(4) 270-294; Springer-Verlag.
- Hammer, J. and Medjahed, B. (1993); "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems"; Journal for Intelligent and Cooperative Information Systems; 2(1) 51-83; World Scientific.
- Handschuh, S.; Staab, S. and Ciravegna, F. (2002); "S-CREAM - Semi-automatic CREATION of Metadata"; Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, 358-372; Siguenza, Spain.
- Harmo-TEN; "Harmo-TEN"; <http://www.harmo-ten.info>.
- Harmonise; "IMHO - Interoperable Minimum Harmonization Ontology"; <http://www.harmonise.org>.
- Hefflin, J.; Hendler, J. and Luke, S. (2001); "SHOE: A Prototype Language for the Semantic Web"; Linkoping Electronic Articles in Computer and Information Science; 6(3); Linkoping.
- Horrocks, I. (1998); "The FaCT System"; Proceedings of the Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux'98), 307-312; Oisterwijk, Netherlands.
- Janowicz, K. and Riedman, C. (2004); "Bridge-IT Technology Watch Report 4"; D7.2.4; BRIDGE-IT (IST-2001-34386).
- Kahn, L. R. and Hovy, E. H. (1997); "Improving the Precision of Lexicon-to-ontology Alignment Algorithms"; Proceedings of the AMTA/SIG-IL First Workshop on Interlinguas; San Diego (CA), USA.
- Kalfoglou, Y. and Schorlemmer, M. (2003); "Ontology mapping: the state of the art"; The Knowledge Engineering Review; 18(1) 1-31; Cambridge University Press.
- Kang, J. and Naughton, J. F. (2003); "On schema matching with opaque column names and data values"; Proceedings of the ACM SIGMOD International Conference on Management of Data, 205-216; San Diego (CA), USA.
- KAON; "KAON - Karlsruhe Ontology and Semantic Web Workbench"; <http://kaon.semanticweb.net>.
- Karvounarakis, G.; Alexaki, S.; Christophides, V.; Plexousakis, D. and Scholl, M. (2002); "RQL: A Declarative Query Language for RDF"; Proceedings of the Eleventh International World Wide Web Conference, 592-603; Honolulu (HA), USA.
- Klein, M.; Kiryakov, A.; Ognyanov, D. and Fensel, D. (2002); "Ontology Versioning and Change Detection on the Web"; Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), 197-212; Heidelberg.

Klusch, M. (2001); "Information Agent Technology for the Internet: A Survey"; Data & Knowledge Engineering; 36(Special Issue on Intelligent Information Integration) 337-372; Elsevier Science.

Laleci, A.; Kirbas, G.; Kabak, Y.; Sinir, S. and Yildiz A. (2004); "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain"; Submitted to Elsevier Science; Elsevier.

Madhavan, J.; Bernstein, P. A.; Domingos, P. and Halevy, A. (2002); "Representing and Reasoning about Mappings between Domain Models"; Proceedings of the Eighteenth National Conference on Artificial Intelligence, 80-86; Edmonton, Canada.

Madhavan, J.; Bernstein, P. A. and Rahm, E. (2001); "Generic Schema Matching with Cupid"; Proceedings of the 27th Very Large Database Conference, 49-58; Rome, Italy.

Maedche, A.; Motik, B.; Silva, N. and Volz, R. (2002a); "MAFRA - A MAPPING FRAMework for Distributed Ontologies"; Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, 235-250; Sigüenza, Spain.

Maedche, A.; Motik, B.; Silva, N. and Volz, R. (2002b); "MAFRA - A MAPPING FRAMework for Distributed Ontologies in the Semantic Web"; Proceedings of the Workshop on Knowledge Transformation for the Semantic Web at ECAI'2002, 60-68; Lyon, France.

Maedche, A.; Motik, B.; Stojanovic, L.; Studer, R. and Volz, R. (2003); "An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies"; Proceedings of the Proceedings of the WWW 2003, 439-448; Budapest, Hungary.

MAFRA Toolkit; "MAFRA Toolkit"; <http://mafra-toolkit.sourceforge.net>.

MEK; "MEK ontology"; <http://www.mek.fi>.

Miller, G. A.; Beckwith, R.; Fellbaum, C.; Gross, D. and Miller, K. J. (1990); "Introduction to WordNet: An on-line lexical database"; Journal of Lexicography; 3(4) 235-244; Oxford University Press.

Miller, R. J.; Haas, L. M. and Hernández, M. A. (2000); "Clio: Schema Mapping as Query Discovery"; Proceedings of the 26th Very Large Database Conference, 77-88; Cairo, Egypt.

Milo, T. and Zohar, S. (1998); "Using Schema Matching to Simplify Heterogeneous Data Translation"; Proceedings of the 24th International Conference Very Large Data Bases, 122-133; New York (NY), USA.

Mitra, P. and Wiederhold, G. (2001); "An Algebra for Semantic Interoperability of Information Sources"; Proceedings of the 2nd. IEEE Symposium on BioInformatics and Bioengineering, 174-182; Bethesda (MD), USA.

Mitra, P.; Wiederhold, G. and Jannink, J. (1999); "Semi-automatic Integration of Knowledge Sources"; Proceedings of the 2nd International Conference on Information Fusion.

Motik, B.; Maedche, A. and Volz, R. (2003); "A Conceptual Modeling Approach for building semantics-driven enterprise applications"; Proceedings of the First International Conference on Ontologies, Databases and Application of Semantics, 1082-1099; Irvine (CA), USA.

Neches, R.; Fikes, R.; Finin, T.; Gruber, T. R.; Patil, R.; Senator, T. and Swartout, W. (1991); "Enabling technology for knowledge sharing"; AI Magazine; 12(3) 36-56; American Association for Artificial Intelligence.

Nonaka, I. and Takeuchi, H. (1995); "The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation"; Oxford University Press.

Noy, N. F.; Ferguson, R. W. and Musen, M. A. (2000); "The knowledge model of Protege-2000: Combining interoperability and flexibility"; Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, 17-32; Juan-les-Pins, France.

- Noy, N. F. and Musen, M. A. (2000); "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment"; Proceedings of the 17th National Conference on Artificial Intelligence, 450-455.
- Noy, N. F. and Musen, M. A. (2001); "Anchor-PROMPT: Using Non-Local Context for Semantic Matching"; Proceedings of the Workshop on Ontologies and Information Sharing at IJCAI '01, 63-70; Seattle (WA), USA.
- Omelayenko, B. (2002b); "RDFT: A Mapping Meta-Ontology for Business Integration"; Proceedings of the Workshop on Knowledge Transformation for the Semantic Web at ECAI'2002, 76-83; Lyon, France.
- Omelayenko, B. (2002a); "Integrating Vocabularies: Discovering and Representing Vocabulary Maps"; Proceedings of the First International Semantic Web Conference, 206-220; Sardinia, Italy.
- Omelayenko, B. and Fensel, D. (2001); "A Two-Layered Integration Approach for Product Information in B2B E-commerce"; Proceedings of the Second International Conference on Electronic Commerce and Web Technologies, 226-239; Munich, Germany.
- Ontobroker; "Ontobroker"; http://www.ontoprise.de/products/ontobroker_en.
- OntoMapper; "ONTOMAPPER - Ontology Automatic Mapping"; http://www.gecad.isep.ipp.pt/GECAD_EN/projectos/ontomapper.htm.
- Pan, J. and Horrocks, I. (2003); "RDFS(FA) and RDF MT: Two Semantics for RDFS"; Proceedings of the Second International Semantic Web Conference, 30-46; Sanibel Island (FL), USA.
- Park, J. Y.; Gennari, J. H. and Musen, M. A. (1998); "Mappings for Reuse in Knowledge-based Systems"; Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management; Banff, Canada.
- Pinto, H. S.; Gómez-Pérez, A. and Martins, J. P. (1999); "Some issues on ontology integration"; Proceedings of the Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends at IJCAI'9., 7.1-7.11; Amsterdam, Netherlands.
- Planserve (2003); "PLANSERVE: Enabling Technologies for Intelligent Planning Services"; submitted to Sixth Framework Programme as an Integrated Project; *waiting for approval*.
- Popa, L.; Velegrakis, Y.; Miller, R. J.; Hernández, M. A. and Fagin, R. (2002); "Translating Web Data"; Proceedings of the 28th Very Large Data-Base Conference, 598-609.
- Rahm, E. and Bernstein, P. A. (2001); "A survey of approaches to automatic schema matching"; The VLDB Journal; 10(4) 334-350; Springer-Verlag.
- Resnik, P. (1999); "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language"; Journal of Artificial Intelligence Research; 11 95-130; AI Access Foundation/Morgan Kaufman.
- Russel, S. and Norvig, P. (1995); "Artificial Intelligence: A Modern Approach"; Prentice-Hall, Inc.
- SANSKI; "SANSKI - Semi-automatic Negotiation Service for Knowledge Interoperability"; http://www.gecad.isep.ipp.pt/GECAD_EN/projectos/sanski.htm.
- Santos, J. and Staab, S. (2003); "FONTE: Factorizing ontology engineering complexity"; Proceedings of the International Conference On Knowledge Capture , 146-153; Sanibel Island (FL), USA.
- Sarini, M. and Simone, C. (2002); "The Reconciler: supporting actors in meaning negotiation"; Proceedings of the Workshop on Meaning Negotiation (Mean-02) at AAAI-02; Edmonton (Alberta), Canada.
- Satine; "Satine: Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer Networks"; <http://www.srdc.metu.edu.tr/webpage/projects/satine>.
- SBO; "Semantic Bridge Ontology"; <http://cvs.sourceforge.net/viewcvs.py/mafra-toolkit/src/pt/ipp/ise/pt/gecad/mafra/sbo/model/res/bridges.rdfs>.

- Sheth, S. A. and Larson, J. A. (1990); "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases"; ACM Computing Surveys; 22(3) 183-236; ACM Press.
- SIGRT; "Sistema de Informação de Gestão de Recursos Turísticos (SIGRT)"; <http://www.dgturismo.pt/irt>.
- Silva, N. (2002b); "Interoperabilidade baseada em conhecimento"; SANSKI-PR-2002-02; GECAD-ISEP-IPP.
- Silva, N. (2002a); "Descrição das evoluções tecnológicas na partilha de conhecimento, e os desafios colocados na sua introdução"; SANSKI-PR-2002-01; GECAD-ISEP-IPP.
- Silva, N. (1998); "Sistemas Holónicos de Produção - Especificação e Desenvolvimento"; Master dissertation; Faculty of Engineering, University of Porto; Porto, Portugal.
- Silva, N. (2003); "Analysis and definition of technological support for ontology mapping"; SANSKI-PR-2003-01; GECAD-ISEP-IPP.
- Silva, N. and Ramos, C. (1999); "Holonc Dynamic Scheduling Architecture And Services"; Proceedings of the International Conference on Enterprise Information Systems; Setúbal, Portugal.
- Silva, N. and Rocha, J. (2004a); "Multi-Dimensional Service-Oriented Ontology Mapping"; International Journal of Web Engineering and Technology;(accepted for publication); Inderscience Publishers.
- Silva, N. and Rocha, J. (2002); "Merging Ontologies using a Bottom-up Lexical and Structural Approach" Challenges in Knowledge Representation and Organization for the 21st Century. Integration of Knowledge across Boundaries. Seventh International ISKO Conference; Ergon Verlag.
- Silva, N. and Rocha, J. (2003a); "MAFRA – An Ontology MAPPING FRamework for the Semantic Web"; Proceedings of the 6th International Conference on Business Information Systems; Colorado Springs (CO), USA.
- Silva, N. and Rocha, J. (2003b); "MAFRA – Semantic Web Ontology MAPPING FRamework"; Proceedings of the Seventh Multi-Conference on Systemics, Cybernetics and Informatics; Orlando (FL), USA.
- Silva, N. and Rocha, J. (2003c); "Ontology Mapping for Interoperability in Semantic"; Proceedings of the International Conference WWW/Internet 2003; Algarve, Portugal.
- Silva, N. and Rocha, J. (2003d); "Semantic Web Complex Ontology Mapping"; Proceedings of the Web Intelligence 2003, 82-88; Halifax, Canada.
- Silva, N. and Rocha, J. (2003e); "Service-Oriented Ontology Mapping System"; Proceedings of the Workshop on Semantic Integration of the International Semantic Web Conference; Sanibel Island (FL), USA.
- Silva, N. and Rocha, J. (2004b); "Semantic Web Complex Ontology Mapping"; Web Intelligence and Agent Systems Journal; 1(3-4) 235-248; IOS Press.
- Silva, N.; Rocha, J. and Cardoso, J. (2003); "E-Business Interoperability through Ontology Semantic Mapping" Processes and Foundations for Virtual Organizations, IFIP TC5/WG5.5 Fourth Working Conference on Virtual Enterprises; 315-322; Kluwer.
- Silva, N.; Santos, J. and Rocha, J. (2004); "Proposal for the combination of ontology assemble and ontology mapping processes"; Proceedings of the International Conference on Knowledge Engineering and Decision Support; Porto, Portugal.
- Sinir, S.; Yildiz A.; Kirbas, G. and Gurcan Y. (2004); "Semantically Enriched Web Services for Travel Industry"; Submitted to SIGMOD Record; ACM.
- Sintek, M. and Decker, S. (2002); "TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web" International Semantic Web Conference (ISWC-2002); 364-378; Springer.

- Sowa, J. F. (1999); "Knowledge Representation: Logical, Philosophical, and Computational Foundations"; Brooks Cole Publishing Co.
- Stojanovic, L.; Maedche, A.; Motik, B. and Stojanovic, N. (2002a); "User-driven Ontology Evolution Management"; Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, 197-212; Heidelberg.
- Stojanovic, N.; Stojanovic, L. and Volz, R. (2002b); "A reverse engineering approach for migrating data-intensive web sites to the Semantic Web"; Proceedings of the Intelligent Information Processing, World Computer Congress 2002; Montreal, Canada.
- Stuckenschmidt, H. and Klein, M. (2004); "Towards Automatic Partitioning of Class Hierarchies"; Proceedings of the International Conference on Knowledge Engineering and Decision Support, 287-294; Porto, Portugal.
- Stuckenschmidt, H. and Visser, U. (2000); "Semantic Translation Based on Approximate Re-Classification"; Proceedings of the Workshop Semantic Approximation, Granularity and Vagueness at KR 2000; Breckenridge (CO), USA.
- Stuckenschmidt, H. and Wache, H. (2000); "Context Modeling and Transformation for Semantic Interoperability"; Proceedings of the International Workshop Knowledge Representation meets Databases at ECAI'2000, 115-126; Berlin, Germany.
- Stuckenschmidt, H.; Wache, H.; Voegelé, T. and Visser, U. (2000); "Enabling technologies for interoperability"; Proceedings of the Workshop on 14th International Symposium of Computer Science for Environmental Protection, 35-46; Bonn, Germany.
- Studer, R.; Benjamins, R. and Fensel, D. (1998); "Knowledge Engineering: Principles and Methods"; Data & Knowledge Engineering; 25(1-2) 161-197; Elsevier.
- Stumme, G. and Maedche, A. (2001); "Ontology Merging for Federated Ontologies on the Semantic Web"; Proceedings of the IJCAI'01 Workshop on Ontologies and Information Sharing, 91-99; Seattle (WA), USA.
- Sure, Y.; Erdmann, M.; Angele, J.; Staab, S.; Studer, R. and Wenke, D. (2002); "OntoEdit: Collaborative Ontology Development for the Semantic Web"; Proceedings of the First International Semantic Web Conference, 221-235; Sardinia, Italia.
- Sycara, K.; Lu, J. and Klusch, M. (1998); "Interoperability among Heterogeneous Software Agents on the Internet"; CMU-RI-TR-98-22; Carnegie Mellon University.
- TIS; "tiscover"; <http://www.tiscover.com>.
- TourinFrance; "TourinFrance"; <http://www.tourisme.gouv.fr>.
- UML; "Unified Modeling Language"; <http://www.uml.org>.
- Uschold, M. and Jasper, R. (1999); "A Framework for Understanding and Classifying Ontology Applications"; Proceedings of the Workshop on Ontologies and Problem-Solving Methods at IJCAI99; Stockholm, Sweden.
- Uschold, M.; King, M.; Moralee, S. and Zorgios, Y. (1998); "The Enterprise Ontology"; The Knowledge Engineering Review, Special Issue on Putting Ontologies to Use; 13(1) 31-89; Cambridge University Press.
- van Elst, L. and Abecker, A. (2002); "Negotiating Domain Ontologies in Distributed Organizational Memories"; Proceedings of the AAAI-02 Workshop on Meaning Negotiation (MeaN-02) held in conjunction with Eighteenth National Conference on Artificial Intelligence, 32-35.
- Visser, P. R. S.; Jones, D. M.; Bench-Capon, T. J. M. and Shave, M. J. R. (1997); "An Analysis of Ontological Mismatches: Heterogeneity versus Interoperability"; Proceedings of the AAAI Spring Symposium on Ontological Engineering; Stanford (CA), USA.

Wache, H.; Voegele, T.; Visser, U.; Stuckenschmidt, H.; Schuster, G.; Neumann, H. and Huebner, S. (2001); "Ontology-based integration of information - a survey of existing approaches."; Proceedings of the Workshop on Ontologies and Information Sharing of the International Joint Conference on Artificial Intelligence, 108-117; Seattle (WA), USA.

Webster; "Merriam-Webster Online Dictionary"; <http://www.webster.com>.

Wiederhold, G. and Genesereth, M. R. (1995); "The Basis for Mediation"; Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS-95), 140-157; Vienna, Austria.

Wiig, K. M. (2000); "The Intelligent Enterprise and Knowledge Management" UNESCO's Encyclopedia of Life Support Systems.

WoW; "WhatsOnWhen"; <http://www.whatsonwhen.com>.

Xiao, H.; Cruz, I. and Hsu, F. (2004); "Semantic Mappings for the Integration of XML and RDF Sources"; Proceedings of the Workshop on Information Integration on the Web; Toronto, Canada.

XQuery; "XQuery 1.0: An XML Query Language"; <http://www.w3.org/TR/xquery>.

XSLT; "XSL Transformations"; <http://www.w3.org/TR/xslt>.