Revised Ontology Task Handbook

Deliverable TONES-D07

Enrico Franconi[1] (editor),
Sergio Tessaris[1], Bernardo Cuenca Grau[2], Boontawee Suntisrivaraporn[3],
Carsten Lutz[3], Ralf Möller[4], Domenico Lembo[5]

[1] Free University of Bozen-Bolzano
[2] The University of Manchester
[3] Technische Universität Dresden
[4] Technische Universität Hamburg-Harburg
[5] Università di Roma "La Sapienza"

Project:	FP6-7603 – Thinking ONtologiES (TONES)
Workpackage:	WP1 – Assessment of fundamental ontology-based tasks
Lead Participant:	Free University of Bozen-Bolzano
Reviewer:	Maurizio Lenzerini
Document Type:	Deliverable
Classification:	Public
Distribution:	TONES Consortium
Status:	Final
Document file:	D07_RevisedOntologyTaskHandbook.pdf
Version:	2.2
Date:	4 August, 2006
Number of pages:	56

Abstract

Based on the review of the state of the art (Deliverable D1), suitable reasoning and meta-reasoning mechanisms associated to the processing of ontologies have to be identified. These mechanisms have to be classified according to their applicability in the various user activities in which ontologies are dealt with.

This version of the ontology task handbook contains the revised proposal for ontology-based use cases and the associated reasoning and meta-reasoning services, to be used in the project.

The set of ontology-based tasks underlying the identified use cases lays the basis for the whole project. The central role of these tasks makes it necessary to assess and validate their suitability in an early stage. For this purpose the selected tasks have been tested on small examples that are chosen so that the results of their application can easily be matched with the expected outcome. Based also on the inputs of the first industrial advisory team workshop (WP8), the set of ontology-based tasks have been finalised.

Document Change Record		
Version	**Date**	**Reason for Change**
v.2.0	3 July, 2006	Final public draft
v.2.1	21 July 2006	Version ready for internal review
v.2.2	11 August 2006	Final version

Contents

1 Introduction

It is well known that a major barrier between industry and academia is that the former speaks in terms of problems and solutions while the latter in terms of technologies and research issues. A use case is basically a story that relates a problem to a solution and a solution to a technology, and as a consequence to a research issue – and in our project this latter is an *ontology-based task*. Use cases are thus good language tools to facilitate understanding between an organisation and a research group since they set a common context and simple natural language for interaction.

In order to support the communication between academia and industry to their mutual benefit, the TONES project has formed an industrial advisory team, which is providing input and feedback throughout the project. The team currently lists members of major industrial players both in Europe and worldwide: Bell Labs, IBM, Sun Microsystems; as well as some of the more innovative technological startups in this area: webXerpt (Germany), Racer-Systems (Germany), Network Inference (USA). The project's industrial advisory team includes some of Europe's leading providers and users of Web technology. As well as keeping the consortium informed about current industrial practice and requirements, they provide test sites, case studies and additional exploitation pathways. A clear case of establishing mutual benefit for both groups is in the exchange of experience and knowledge, from the industry partners that of actual business operations and requirements, and from academia the current state of the art in ontology research. The aim of this exchange will be that academia can be better aware of industrial needs and hence direct research in the ontology field towards those needs, and that industry can then consequently benefit from research efforts tailored to the needs of their business.

This document acts as an informative input to the scientific research within TONES. In order to provide an authoritative guideline for research the collection of cases have been subject to an analysis of the intended solutions for which ontology-based technologies are being identified and the technological locks preventing those solutions. In June 2006 a workshop has been organised, in which the members of the industrial advisory team have been invited to discuss, providing useful inputs, the definition of the ontology related tasks associated to the user activities (see the TONES deliverable D04 including the minutes). This has led lead to the production of this revised ontology task handbook. At month 30, in a second workshop, the industrial advisory team will provide more useful and up-to-the-point feedback on the preliminary results of the project.
Surprisingly enough, the advisory team members in the first workshop considered relevant for their advanced applications most of the main topics highlighted in the draft version of this deliverable. The variability of their requests and comments gives a solid foundation to the general and abstract framework that we are considering in TONES. So, in the revised version we essentially confirmed the topics as in the draft, and we could now also be more specific about the research challenges that the project will study for the next period.

The work of collecting, analysing, and categorising ontology-based user activities is

not stemming from nowhere. We are benefiting from the participation of some of the scientific partners of TONES in current projects within EU Network of Excellences (NoE): KnowledgeWeb ("Realizing the Semantic Web", EU-IST Network of Excellence IST-2004-507482), Rewerse ("Reasoning on the Web with Rules and Semantics", EU-IST Network of Excellence IST-2004-506779), and Interop ("Interoperability Research for Networked Enterprises Applications and Software", EU-IST Network of Excellence IST-508011).

Starting from the up-to-date and exhaustive analysis of the state of the art in ontology-based technologies done as first step in our TONES project [3] – which was structured driven by the technology rather that by the use cases – much of this document is organised according to the structure and the kind of motivations of the *use cases* deliverables of the abovementioned Networks of Excellence. In particular, we have considered the lessons learnt from the deliverable [10] "Prototypical Business Use Cases" from the KnowledgeWeb NoE. This document was created with a close cooperation between the industry partners and the research partners, and it went through several stages of feedback among the two parties. We will adopt a similar structure in the description of the scenarios, but we will do more. Remember what we were saying at the beginning: we will tell stories that relates a problem to a solution and a solution to a technology, and as a *final* consequence to a research issue. As a matter of fact, the missing aspect of a big network such as the KnowledgeWeb NoE (but we have found a similar problem in the two other NoEs we are part) is the final connection with challenging research issues, and a structuring of the analysis based on concrete technologies that can provably give an added value to the case. This is mostly due to the nature itself of such networks, which have as one of their goals to make possible and to support more basic research oriented projects such as TONES. In this perspective, we have to say that the concrete choice of the use cases in KnowledgeWeb [10, 13, 12], Rewerse [5, 2], and Interop [4], while it has been organised in interesting grids, it is not homogeneous and not strictly tied with the innovative technologies and scientific advancement we want to deal in a project like TONES.

In the following, the structuring grid of scenarios will be introduced, and the use cases will be presented organised within this grid. At the end of this document, four concrete example use scenarios will be thoroughly analysed.

2 A Framework of Ontology-based User Activities

In our analysis, we will focus on the notion of ontologies as means to organise *information*, which is the paradigm characterising most of the ontology-based applications we have analysed in [3]. This is supported in principle by applying the classical analysis of Russell Ackoff [1], who considers knowledge (and therefore its technological counterpart in our context: ontologies) as being part of the so called "Information Hierarchy".

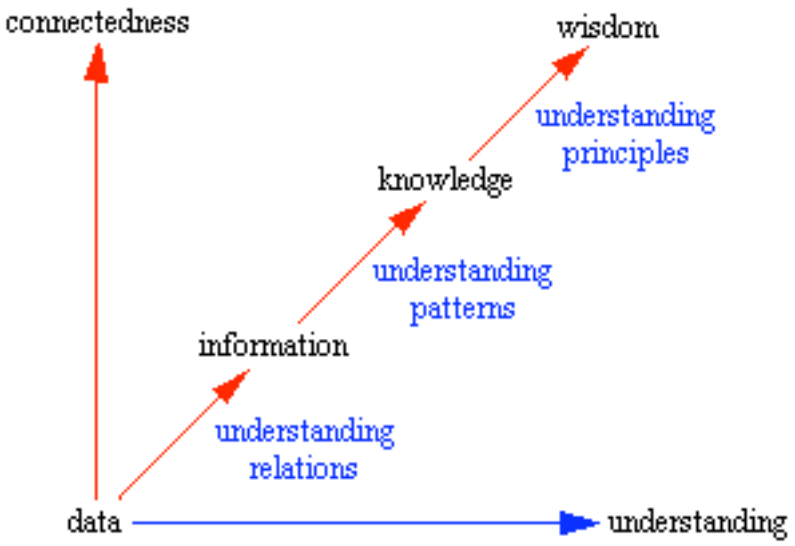

Figure 1: The transitions from data, to information, to knowledge, and finally to wisdom.

According to Figure 1, *knowledge* represents the necessary step that mediates between *information* and *wisdom*. Knowledge gives a higher level of connectivity to information, by allowing for the understanding of patterns and behaviours, as opposed to the mere understanding of relations. *Wisdom* is at the top of the information hierarchy, representing the understanding of the principles that organise knowledge. While we organise the ontology-based user activities into categories, we are at the level of *wisdom*, since we actually classify the abstract roles (i.e., the general principles) that knowledge plays with information-based scenarios. So, we will refer to a categorisation framework where ontologies play the role of *conceptual* models of some (semi-)structured information (Figure 2), and in which it is possible that the information and/or the conceptual models may need to be integrated (Figure 3).

Figure 2 depicts the role of an ontology (the blue top layer) wit respect to the information structures (the middle layer) and to data (the bottom layer, modelled itself by the information structures). In this sense, an ontology is a formal conceptualisation of the information world, specifying a set of *constraints* – which declare what should necessarily

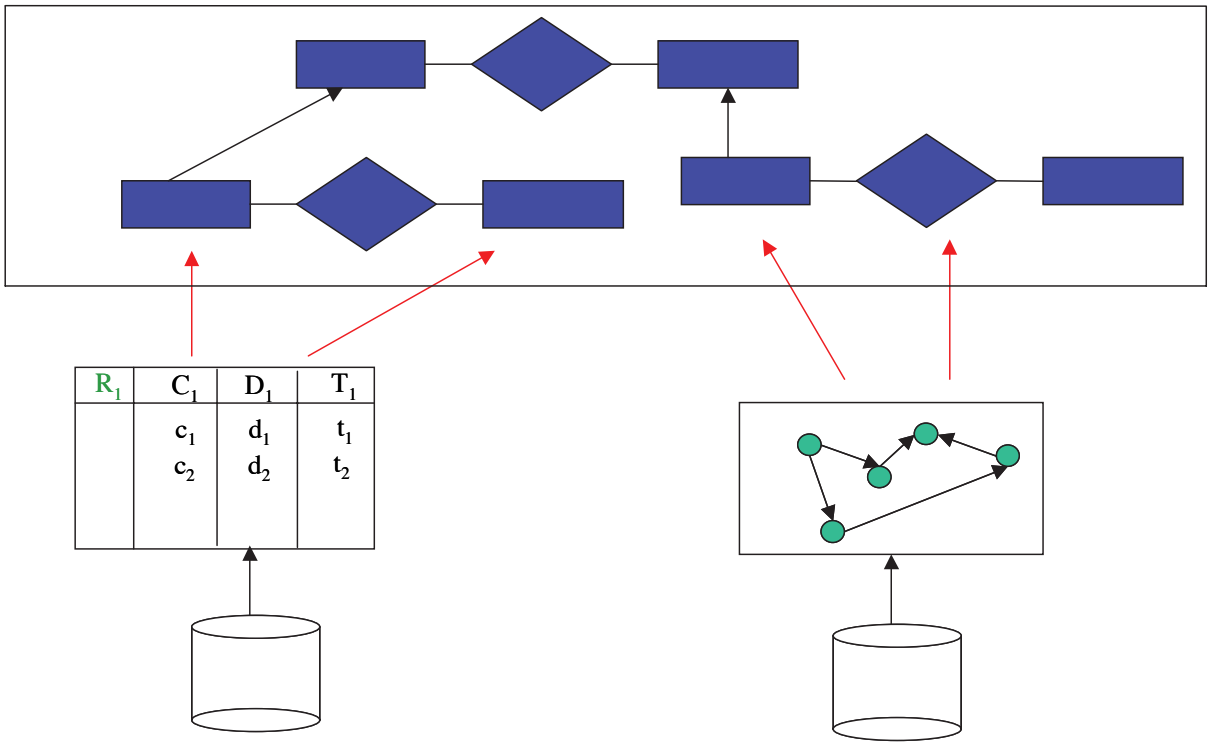

Figure 2: The general framework involving data, information, and knowledge.

hold for any data organised according to that information structure. In general, several kinds of information structures may be considered: relational databases, web pages, XML semi-structured information, etc. In this framework, several important scenarios are conceivable.

The first obvious activity we can consider is the *ontology-as-object* activity. Within this category – where scenarios considering ontologies as their ultimate object of analysis and use – we can analyse the life-cycle of the ontology itself: there is the creation of the ontology, the maintenance of the ontology, the deployment of the ontology as an object. The creation and the maintenance of the ontology may be conceived also in presence of proper data: for example, this would be the case of bootstrapping the design of an ontology from the available database schema, through a generalisation process. As a matter of fact ontology modelling deals with the question on how to describe in a declarative and reusable way the domain information of an application, its relevant vocabulary, and how to constrain the use the data, by understanding what can be drawn from it. The deployment of ontologies in *ontology-as-object* activities is typically an operation of navigation or intelligent querying of the ontology alone; this knowledge exploration activity is the kind of activity that is needed for example when classifying external documents or records.

More challenging are the activities that involve an ontology together with its information source. Here, typical problems are the verification of the consistency of the data with respect to the knowledge, and most notably the *access to the information* mediated

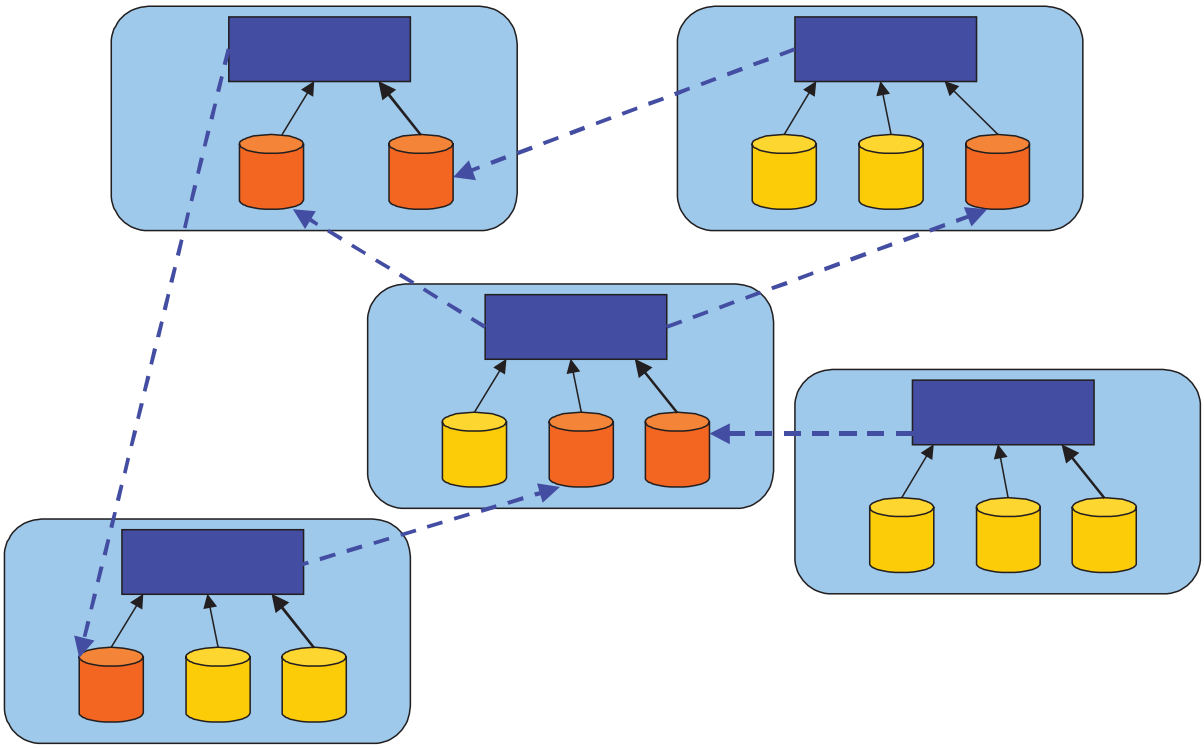

Figure 3: The general framework involving integration.

by the ontology. Good ontologies put their emphasis on the correct and semantically rich representation of complex properties and relations that may exist in the data; they should allow for an abstract representation of information which resembles the way they are actually perceived and used in the real world, thus shortening (with respect to the more traditional data models) the semantic gap between the domain and its representation. With this perspective in mind, the user would prefer to query the information system using the richer vocabulary of the ontology. The vocabulary of the information structure could be seen in turn either as a subset of the conceptual vocabulary – this is the simplistic view – or more generally as a set of (materialised) views over the vocabulary of the ontology. However, in this case we have to solve the problem of view-based query processing. The problem requires to answer a query posed to a database – the one defined by the ontology – only on the basis of the information in a set of (materialised) views, which are again queries over the same database. In the process, the information contained in the ontology should be of course taken into account. In this framework, more subtle problems come into play, since now the complexity of the scenario involves also the dimension of the data, which usually is much bigger than the dimension of the knowledge.

Figure 3 shows an extension of the basic framework, that considers the ontology as part of a larger context: what happens when several ontologies within several information systems do interact. Again, we can conceive *ontology-as-object* activities, whose purpose

is to (virtually) derive a sort of *unified* ontology out of several (possibly distributed) ontologies. But the real challenge as it was posed by the semantic web community recently is the problem of *information integration mediated by ontologies*. Several approaches have been proposed; for example, a popular way of solving the integration problem recently is based on peer-to-peer techniques, which has several computational advantages over more classical integration solutions.

Another important dimension is associated to the *type of information* that is characterised by the ontology: for example information about states, about time, about space, about actions or events. This requires special constructs to be available in the ontology language, that could capture the semantic richness of the different information types. From the applicative point of view, it is relevant to have the ability to represent this kind of information, both in the case of the ontology-as-object scenarios and in the information access scenarios.

3 The Ontology-based User Activities

In this Section we introduce few ontology-based scenarios, chosen for their representativeness both in the applicative world and in the research world. For each of those scenarios, we will introduce a well worked out use case, which emphasises the interconnectedness:

"problem" ⇒ "solution" ⇒ "technology" ⇒ "research issue".

Each ontology-based activity will be summarised with the following structure:

Challenge
A brief description of the activity high level advanced goals.

Key benefits
Why this activity is relevant and which are its benefits to which community.

Solution
How the activity is meant to be solved, based on ontology-based technologies.

Added value
Why ontologies play a crucial role in the solution of the activity.

Technology
The ontology-based technologies at the heart of the envisaged solution to the activity.

Ontology-based Research Tasks
The key research challenges involved in the ontology-based technologies.

Please note that this document is organised according to the scenarios how they are perceived from a user point of view. These user activities will be analysed according to specific solutions based on ontology reasoning. The ultimate purpose of this document is to justify the ontology-based tasks that are crucial – i.e., provide added value – in the solution of the use cases. The study of those ontology-based tasks will be the focus of the research of the TONES project.

3.1 Ontology Design and Maintenance

Challenge
A methodology and a tool to design and maintain ontologies.

Key benefits
Higher quality ontology design; bootstrap from information system; support of unskilled user.

Solution
Interactive design process via feedback and refinement; views with conceptual model; ontology customisation.

Added value
It goes beyond a graphical front-end for ontology design computing only classifications of concepts.

Technology
Several nonstandard reasoning tasks are needed.

Ontology-based Research Tasks
Relationships with database data models, derivation of strictest constraints, reasoning modulo inexpressive languages, nonstandard description logic reasoning.

In this section, we introduce ontology-based tasks important for the design and maintenance phase of ontology engineering. Motivated by problems encountered in industrial applications, we will describe scenarios in which those problems occur. Each problem scenario depicts sub-tasks of the ontology-based design and maintenance task, and describes the needs of those sub-tasks in ontology engineering. The necessity to overcome those problems and to solve those identified sub-tasks call for key challenges that are crucial to solve those application problems. We will propose a number of solutions to the identified sub-tasks. These may be based on existing ontology-based technologies or may require innovation, both of which are to be investigated and studied throughout the TONES research project.

3.1.1 Challenge

The key role of ontologies today, both in information management in general and the Semantic Web in particular, has led to the rapid development of a large number of ontologies. These ontologies have, however, usually been developed in an ad-hoc manner by domain experts, often with only a limited understanding of the semantics of ontology languages. The result is that many ontologies are of low quality—they make poor use of the languages in which they are written and often do not accurately capture the author's rich knowledge of the domain. To alleviate this problem, a good ontology building methodology has to be devised, and tools supporting the task of ontology engineering must be available to domain experts. Such a methodology should at least exploit the support provided by state-of-the-art logic-based ontology languages and also facilitate the experts to take advantage of the full expressive power of the languages. To the best of our knowledge, such a methodology and tools that help an expert design and maintain his/her ontologies with the required support are still missing.

In order to overcome these problems, ontology engineers need a methodology and tools for building high quality ontologies and for retaining that high quality throughout the ontology life-cycle. Indeed, the growing importance of ontologies in various application domains calls for the design and development of the basic infrastructure for building, merging, and maintaining ontologies. However, there is still a fundamental lack of basic technical and methodological means for developing high quality ontologies and for preserving the quality and reusability of ontologies over time.

Such a methodology is by all means needed within an ontology design tool, in order to build high quality ontologies since (*i*) domain experts are, in general, not experts in the ontology language they are working with, (*ii*) formalising one's knowledge in any kind of formalism is, in general, a highly complicated task which requires a lot of discipline, perseverance, and/or support. These problems are often aggravated by the fact that one ontology is built by several domain experts, and that ontologies need to be maintained, extended, and most importantly integrated/interoperated over time.

Other limitations of the current methodologies arise from the current status of the reasoning techniques employed by the state-of-the-art ontology systems. First of all, the expressive power required for representing high-quality ontologies is still not fully provided. Secondly, the sheer size of realistic ontologies pose problems that may require new optimisation techniques for reasoning. Thirdly, the fact that ontologies need to be integrated and interoperated requires the investigation of novel paradigms and of reasoning problems tailored towards these tasks.

As mentioned above, the ultimate goal of ontology design and maintenance is to provide domain experts with a methodology and tools that help develop high quality ontologies. In order to ensure that an ontology will be of high quality and to ensure that the high quality will be consistently retained, the process of ontology design and maintenance has to be looked into. By doing so, the relevant problems and sub-tasks can be identified. Since designing, together with maintaining, ontologies is a highly iterative process, it usually involves several sub-tasks in possibly different circumstances. We argue that

it is helpful to identify all, or at least as many as possible, such relevant sub-tasks, and then compare them. Some of the sub-tasks might seem very closely related but could lead to different challenges and therefore require different technologies, while some others are drawn from completely distinct scenarios but turn out to call for the same challenge.

In the following, we look into the process of ontology design and maintenance by describing the scenarios of the specific sub-tasks, identifying the key challenges, and also proposing possible solutions to deal with the sub-tasks.

Ontology design is at the very first phase of the ontology life-cycle. A user usually starts with a minimal ontology (possibly empty) and adds some new information. The new information—that is, additional classes or properties on classes in which the user is interested—is supposed to be modelled conceptually. When a new ontology is being modelled from scratch, i.e., no information has been formalised previously, the ontology developer usually defines new classes of interest, as well as properties on them. Due to the lack of guidance of previously well-defined classes, it is likely that the information formalised from scratch is not as intended; for instance, some classes could be inconsistent or a class might unintentionally be subsumed by another one. This process may be bootstrapped by logical automatic reasoning that helps detect all inconsistencies in the ontology and make explicit all (implicit) consequences of facts that have been modelled so far. The most important kind of implicit consequences is classification link, i.e. generalisation/specialisation relationship on classes in an ontology. This is essential for ontology design, since an ontology designer always needs to know whether his/her new classes are correctly related to other classes based on his/her knowledge about the domain. Therefore, besides the discovery of inconsistencies, an ontology-design support tool must be able to automatically compute the classification links of an ontology. This supports the designer during the conceptual modelling process in understanding whether the ontology developed so far actually captures the intended meaning. This either increases the belief in the correctness, completeness and interpretability of the ontology, or gives an argument for its inconsistency.

In contrast to the top-down approach, bottom-up modelling generates new classes from examples. In some cases, it might not be so clear from the start to conceive what classes should be defined in an ontology. However, an expert may know—from his/her knowledge about the domain—that in his/her domain of interest there are instances with some common properties. He/She may also want to define a new class that has exactly these properties and that acts as an abstract concept to those instances. Extracting the commonalities from examples and generating an abstract class for them is, notwithstanding, a complex task that domain experts requires support from ontology reasoners. A tool should then be able to support this modelling task by, given a number of examples, generating a specialised common class having all the input examples as its instances.

In the first two approaches, new classes are modelled without any help from the

existing information in the ontology. It is sometimes more reasonable, especially when the ontology under development is becoming larger, to reuse (part of) existing class description to define a new class, provided that the two classes in consideration have some similarities. In addition, the reuse of old information tends to yield a more accurate and less redundant ontology. Consider the situation where the ontology designer wants to add a new piece of information, say, a new class about the domain, and they also want to define it in an analogous manner to the existing class if any. For instance, the developer may consider adding the class 'Aunt' which is—based on the knowledge of experts about the domain—analogous to the class 'Uncle', which is already in the ontology. Then, he/she will first look up the definition of 'Uncle' in order to formulate the notion of 'Aunt' in a parallel fashion. To support this approach of ontology design, a tool should be able to display and compare certain aspects of classes. In our example, the most notable aspect of the class 'Uncle' that is similar to the class 'Aunt', and thus can be reused in when formulating the latter class, is that uncles (as well as aunts) have a niece or a nephew.

Ontology structuring is an important process that ensures the quality of ontologies. As illustrated in the previous chapter, ontologies are beyond information and data in the "Information Hierarchy", as they are not only concerned with separated pieces of information or data but also the relations among them. Here, *ontology structure* refers to how these relations are defined and organised in an ontology itself. The structure of an ontology is one of the key criteria of good ontology design, since an ontology without a decent structure (i.e., it contains no good relations of pieces of information) is not desired. The most important relationship that defines the ontology structure is the so-called IS-A relationship, i.e., the generalisation/specialisation relationship between classes defined in an ontology. This relationship tells us which class is more general/specialised than which class. Consider an ontology with a very broad and shallow structure in which there are many thousands of IS-A links from a general class to its direct sub-classes. All the direct sub-classes are only known to be a specialisation of the same general class, but they are not connected. We cannot derive any more information from this broad ontology about which ontology classes among the many thousands are more connected to the others. Additionally, it is more exhaustive, and thus more expensive, to navigate shallow-structured ontologies. For example, let us conceive the circumstance in which a user wants to retrieve all the classes to which a particular instance belongs. With a shallow and broad structured ontology, we would end up with instance checking against most classes, i.e., checking whether the instance in question belongs to each class individually. With a better structured ontology, however, we could perform only a few such tests with the help of its good generalisation/specialisation hierarchy, i.e., in case the test is positive for a more specialised class, we can derive the more general classes as well without needing to check them. This means that we consider a situation where an ontology has a broad and shallow structure and does not contain much information about the domain. In this situation, it is always more preferable to

refine the structure of the ontology by introducing new intermediate classes with partial commonalities. These intermediate classes can be understood as *views* that are specifically defined for a number of sub-classes. The domain experts may well know the drawbacks of this kind of structure and may want to improve the structure of their ontology. Unfortunately, it is sometimes too complicated for the domain experts to make this happen, since (*i*) in the domain context in question, it can be counterintuitive to add an intermediate class to the existing hierarchy, (*ii*) even one can come up with an abstract intermediate class that is supposed to generalise a number of sub-classes, it is hard to formally extract the commonalities of these sub-classes, and again (*iii*) this is aggravated by the fact that the domain experts are not experts in the ontology language. The ontology building tools should be able to support this sub-task by providing the ontology experts with the means to introduce an intermediate class given a number of sub-classes. When more than one candidate for being the new class are possible, the tools should present them to the domain experts to let them decide which one is best suited the existing ontology.

We now consider another kind of flaw in ontologies—redundancy—which is also a key criterion important for ensuring the ontology quality. Redundancy can be in the form of classes that are always equivalent either intended or not intended by the designer. According to the scenario depicted above, equivalent classes have IS-A links to each other in both direction, though they do not informatively say anything more about the domain. In other words, redundancy increases the ontology complexity but does not increase information the ontology contains. For this reason, redundancy is often not intended by domain experts, and could very well be an indication of flaws in the ontology. One redundancy may induce another, i.e., different equivalent classes may be used to define or constrain different other classes, resulting in (implicit) equivalent classes. This can be hidden deeply in an ontology that is difficult and often impossible to be found out manually. As a result, the experts need reasoning support to locate sources of redundancy and to resolve them. Therefore, additional sub-task that should be realized is the reduction of redundancy in an ontology by discovering equivalent classes, reusing classes, and to refining the class description.

Ontology customisation is a specific sub-task of the ontology design and maintenance task occurring in a specific situation depicted as follows. Here, we distinguish between ontology engineers and ontology users. The ontology engineer uses the full power of the ontology language to build large and sophisticated ontologies. The ontology user "buys" such ontologies but needs to adapt them to his/her own needs. Since the user is not an expert in ontology languages, he/she uses only a sub-language of the full ontology language, which is easier to comprehend and can be supported by a graphical or frame-like interface. The ontology user also needs more support when defining new concepts, such as the bottom-up construction of ontologies, where concepts are generated from examples; and uses concepts defined in the ontologies he/she bought, but probably without knowing their detailed definitions.

This means that we consider a situation where there is a large background ontology written by an ontology expert in an expressive ontology language, and on top of this an additional user ontology, written in a less expressive language. The user needs reasoning support when building this kind of customised ontologies.

Simplified display is a specific sub-task that helps display relevant information in an easy to understand way to either the developer or the user. One of the criteria for a good ontology design is the uniformity, i.e., classes defined in an ontology should be described in a uniform way. As already discussed in the last paragraph of the 'ontology design' scenario, the designer might want to see a certain aspect of an existing class in order to define a new similar class. In order to make that happen, a tool should be able to extract relevant information drawn from the aspect of interest.

This task is even more important when an ontology is designed by a group of developers, where each expert is responsible for a part of the ontology or where the ontology is being maintained over time by many different developers. A new ontology developer might find it difficult to interpret—to get the intuition behind—a complex class description written in an expressive language. Without a good understanding, he/she might not be able to make use of that piece of information or may do so in an incorrect or unintended way. In this situation, the developer needs a tool that helps reformulate the information in a less expressive language that still captures important aspects. The tool should present an untangled reformulation of the real piece of information to the developers for them to grasp the idea first, while giving also the opportunity to display on demand the full intricate information.

Simplified display can also be employed when an error explanation is given, so that the developer can get the first idea underlying the source of the error, and the full explanation formulated in a more expressive language can be displayed on demand. Additionally, a user graphical front-end tool usually has a frame-like interface which does not support the display of some expressive means used in an ontology. In this setting, a simpler reformulation of a class description can be used to display to the user via such a graphical interface.

Ontology extension is one of the most prominent sub-tasks for the ontology design and maintenance phase. It is actually overlapping with the 'ontology modelling' scenario, since during the modelling process an ontology—which is possibly empty in case that the designer models it from scratch—continuously requires to be extended. But, here we concentrate more on extending an existing ontology which is usually large and well-founded. It is important to note also that this sub-task is very much connected to the task of *ontology merging*, which will be described later in the section 'Ontology Integration'. In fact, both sub-tasks, ontology extension and ontology merging, are concerned with the combination of two or more ontologies; however, there are a few key patterns of use that differentiate one from another. (i) ontology extension usually deals with two ontologies of different sizes, namely, an existing large ontology and a small augmenting ontology. This is in contrast to

the peer-to-peer ontology merging, in which ontologies to be merged are often of the same or comparable size. (*ii*) ontology extension usually gives higher priority to the existing larger ontology than the smaller augmenting one. This means that whenever an inconsistency emerges or the resulting extension is not conservative, the augmenting ontology is the first place to be looked into and to be modified.

In many application domains of ontologies such as the semantic web and bio-medical informatics, ontologies are not static. They evolve over time throughout their life-cycle. The necessity to maintain an ontology, especially when a new piece of information is available and has to be added to the existing ontology, calls for the sub-task of ontology extension. When such operation is performed on a well-established and tested ontology, it is of prime importance for the ontology developer to have control over the consequences of his/her modifications. In particular, it frequently happens that unintended consequences are introduced, and such consequences can be far from obvious. Therefore, it is important that an ontology building tool supports the task of ontology extension. The tool should automatically check whether the resulting extended ontology is conservative with respect to the old one, and in case that it is not a conservative extension, the tool should be able to pinpoint a witness of non-conservativeness.

Error management is as important to ontology design and maintenance as debugging is to software engineering. Errors could be made throughout the task of ontology engineering even by ontology language experts themselves, let alone domain experts who often know only little about the language used. For instance, when modelling an ontology the developers may discover *surplus* consequences that, they know intuitively as domain experts, should not follow from the ontology, or the other way round, some expected consequences may be *missing*. Most available ontology reasoners, which typically offer a set of standard inference services, are able to determine whether or not a particular sentence is a logical consequence of the ontology. However, they simply report entailments or non-entailments without providing any justification (proof or disproof) for them. In this situation, a more powerful tool is necessary. Such a tool should be able to give an explanation to such surplus/missing consequences, i.e., to give reason to *why* those consequences are apparent/missing and to pinpoint *what* causes them so. Nevertheless, having seen the source of errors, an unexperienced developer may or may not tell—based on what he/she has seen—what actions to take to debug the errors. For alleviate this problem, the tool could be topped with the capability to suggest the experts *how* to revise the ontology back to the original consistent state by making as little change as possible. Such a suggestion for revision may be of the form that describes what measures to take to rectify the ontology. It might also display to the experts a number of possibilities to fix the error.

Closely related to the task of 'ontology extension', augmenting an ontology with some new classes or new constraint may introduce inconsistencies. In the 'ontology

extension' scenario, the facts in the existing ontology may be compromised in such a way that the augmented ontology is consistent. However, we assume here that the existing ontology has been well-established and contains no flaws, and that any inconsistencies can only be the result of added information. In this situation, the experts need a tool that support pinpointing the source of inconsistencies possibly by showing to the experts a subset of added information that can still lead to the inconsistencies. The experts can then revise the relevant piece of added information and unravel the conflict.

3.1.2 Added Value

Most of the current ontology design tools are just graphical front-ends, in which the only role of ontology-based reasoning is to identify inconsistent classes—without telling the developer any hints to inconsistency—and to construct specialisation/generalisation hierarchies. None of the existing tools provides a helpful methodology supporting the developer, nor is there an interplay between the various phases of the methodology and sophisticated reasoning tasks driven by the ontology itself. The emphasis will be on a methodology which suggests revisions and progress of the ontology, through reasoning in the background. As we have already illustrated in the previous section, the problem scenarios call for the need of various ontology design and maintenance based tools, many of which can be interoperating or even merged. It is important that the developer knows *what* tool to use to solve a problem at hand; *how* to make use of a particular tool; and, *when* to exploit the combined power of different tools to achieve a certain goal. To this end, a methodology for doing so must be studied and made available.

With the sub-tasks 'simplified display' and 'error management', better and more interactive ontology editing/debugging tools can be realised. The more user-friendly editor and/or debugger not only give developers a sense of control over their modelling and hence their ontologies, but also encourage them to experiment more freely with their ontologies and the ontology language. In addition, by playing and working around with errors and the debugging process, developers may come to understand their ontologies even more.

3.1.3 Technology

The scenarios described above depicts a number of sub-tasks important for the design and maintenance phase of ontology engineering. These sub-tasks are clearly based on various technologies, some of which are already in existence and theoretically well-founded. Some illustrated sub-tasks and solutions, however, need novel technologies and methodologies devised specifically to handle them. The following are some examples of available technologies that could be adopted to our proposed solutions to ontology design and maintenance tasks:

- *standard reasoning tasks* on logic-based ontology languages, such as, the computation of all (implicit) consequences of the ontology, the computation of gener-

alisation/specialisation hierarchy of classes occurring in the ontology, consistency checking, etc.

- *non-standard reasoning tasks*, e.g, the computation of least common subsumers (LCS) and most specific concepts (MSC) can be used to facilitate bottom-up ontology design. In addition, LCS is exactly what we would need for the sub-task of ontology structuring, i.e., one can compute LCSs and add them as new classes to the broad and shallow ontology to help improve its structure. Concept rewriting could probably be integrated in standard ontology reasoners or ontology editors to facilitate the sub-task of simplified display.

- other ontology-based technologies which handle belief revision and/or explanation of consequences (respectively, explanation of missing consequences) are promising candidates for bootstrapping some of the proposed sub-tasks, especially the error management task.

3.1.4 Ontology-based Research Tasks

The ontology-based research tasks that are at the core of the abovementioned technology for the ontology design and maintenance activities are summarised in this Section. A deeper discussion on the tasks, from a technological perspective can be found in the TONES Deliverable D05 "Tasks for Ontology Design and Maintenance".

- *(to support ontology design)* The standard basic reasoning problems, i.e., ontology consistency checking, class satisfiability checking, computation of generalisation/specialisation hierarchy.

  Remember that a concept C1 *is subsumed by* (or, *is-a*) a concept C2 if and only if every instance of C1 is also and instance of C2 – written C1 $\sqsubseteq$ C2. For example, Vene $\sqsubseteq$ Vessel or Heart $\sqsubseteq$ Organ. The subsumption relationship defines the structure of the ontology and it is mainly used for navigating the ontology during browsing.

  In Figure 4 the main window of the Protégé ontology design tool is shown. It is evident that the main parts are based on the subsumption relation between classes: the subclass explorer, the superclass explore, the tracker of the changes of the direct superclasses. The subsumption relation between concepts induces also a taxonomy, which can be represented graphically as a directed graph: the taxonomy is the most concise (and intuitive) representation of the subsumption relation; every ontology design tool provides support for subsumption.

  Other standard basic reasoning tasks that are relevant in ontology design are consistency checking for classes (check whether it is possible that the concept description within the ontology allows for at least an instance - if not, it is clearly a useless concept, being equivalent to the empty class), consistency checking for the ontology itself (check whether some data fulfilling the constraints imposed by the ontology may at all exists), and equivalence/disjointness checking between concepts (check

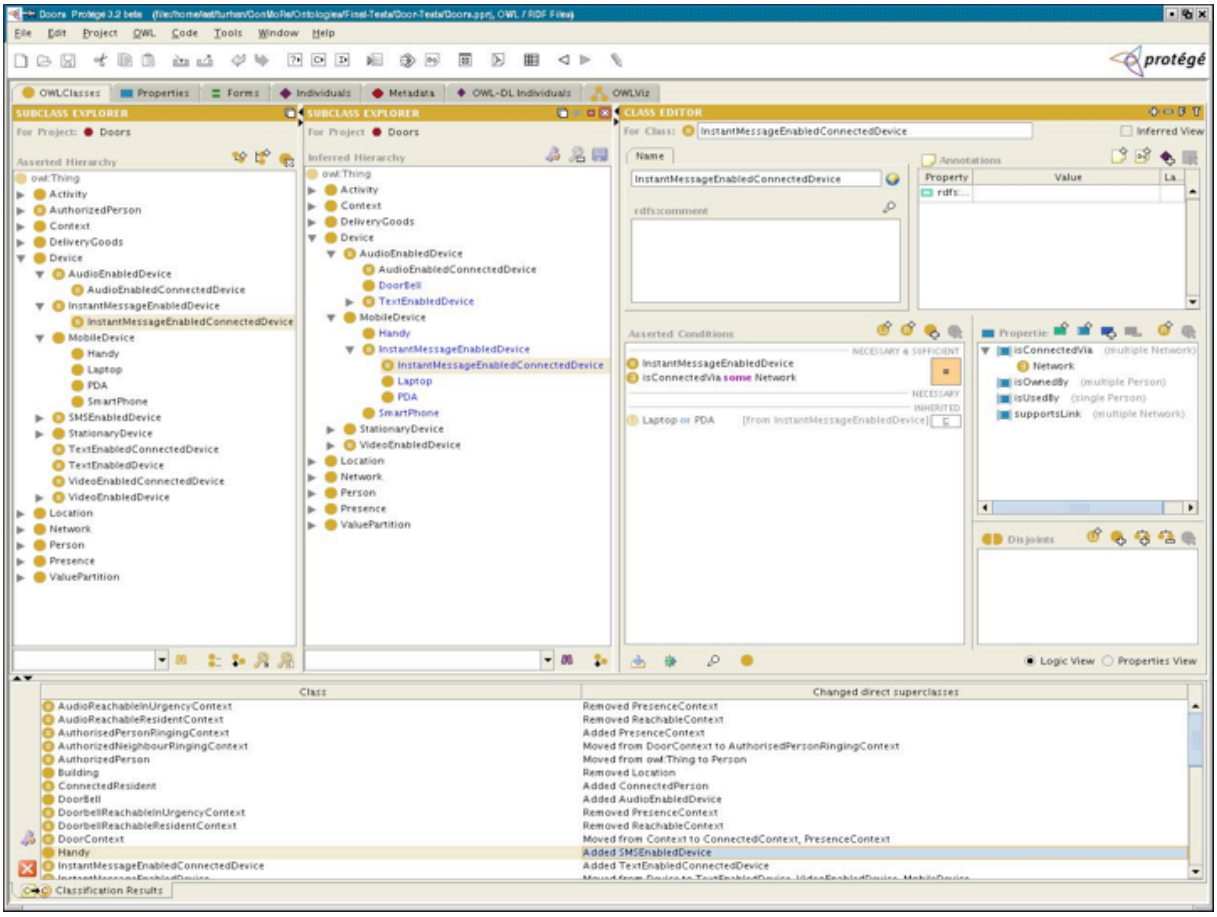

Figure 4: Ontology design with Protégé.

whether two concepts denote the same set of instances or they don't have any instance in common).

It is possible to generalise the kind of ontology-based task needed for ontology design, by considering the derivation of (a finite number of) the most specific basic constraints (corresponding to the atomic constructs available in the ontology language) which are applicable given the ontology. The most important case is the derivation of the most specific cardinality constraints involved in relations between classes. This non-standard reasoning task is already present in some advance ontology design tools, such as Icom (see Figure 5, where the green link is deduced by the tool, as the strictest consequence of the ontology – see the last Chapter of this deliverable for this example fully worked out).

Of course, a research challenge for Tones is to develop correct, complete, and efficient algorithms for the abovementioned basic standard and non-standard reasoning tasks for the different expressive ontology languages at hand.

- *(to support ontology structuring and ontology customisation)* The non-standard rea-

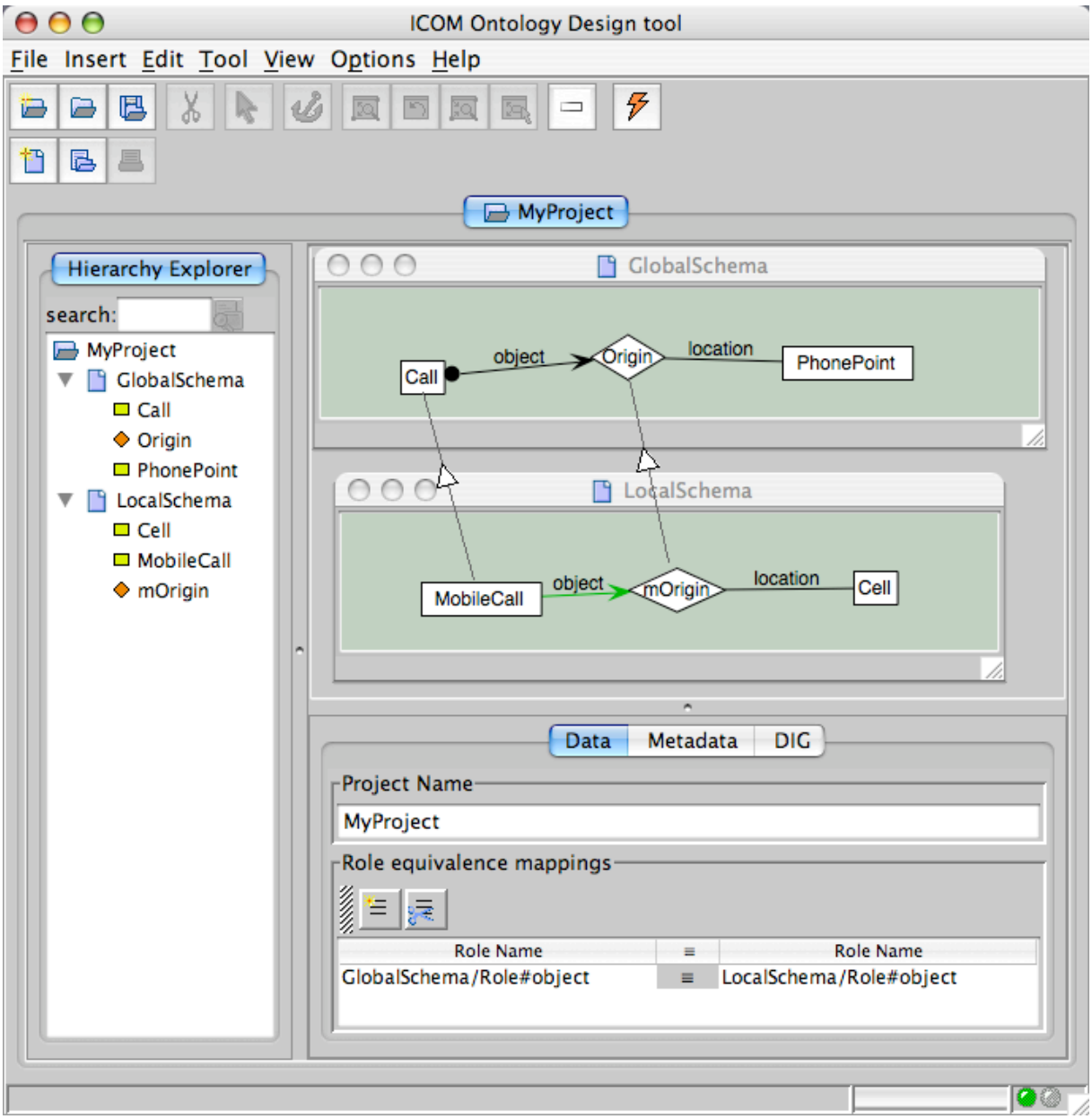

Figure 5: The Icom ontology design tool.

soning problems: the computation of least common subsumers and most specific concepts.

The taxonomic information in the ontology is exploited to structure the evolution of the ontology itself. When authoring a new concept, a designer often knows its intended location in the taxonomy, and instead of writing the concept description from scratch, it may be supported by the system that automatically generates a candidate description for the concept - by proposing something between the least common subsumer of the subclasses of the concept in the taxonomy, and the most

specific subsumee of the superclasses of the concept in the taxonomy; the designer may then manually fine-tune this description. Such candidate description represents the commonalities of the intended subclasses: for example, the concept Vessel may be introduced by first considering the least common subsumer of Vene and Artery.

The reasoning tasks computing the most specific concepts and the least common subsumers are very complex, and it is a research challenge to find effective algorithms that solve the problem for expressive ontology languages. As a matter of fact, the problem is very sensitive to the expressivity of the ontology language: it may vary from being a trivial problem (in case of the presence of disjunction) to be an undecidable one.

- *(to support ontology customisation)* In addition to the previous item, reasoning with two ontologies: one acts as a background knowledge formalised using the full expressive power of the ontology language, and the other is the user ontology formalised in an inexpressive sub-language. The reasoning may use only partial information from the background ontology, e.g., using only the pre-computed generalisation/specialisation hierarchy. The reasoning task associated to this problem is again a non-trivial one.

- *(to support simplified display)* The computation of concept approximation, concept matching and concept rewriting. The study of this tasks is becoming subject of interest of the semantic web community, where the problem is relevant. The Knowledge Web Network of Excellence has an entire workpackage dedicated to this reasoning tasks. Our projects aims at giving more sound foundations to this problems.

- *(to support ontology extension)* New reasoning problems will be involved, some of which could be the key reasoning problems for ontology integration. This may include the problem of deciding conservative extension and the computation of witness concepts for non-conservativeness. The notion of a conservative extension plays a central role in ontology design and integration: it can be used to formalise ontology refinements, safe ontology mergings, and independent modules within an ontology. An extended ontology is a conservative extension if and only if every consequence in the alphabet of the original ontology given the extended ontology is also a consequence of the original ontology alone. Intuitively, a conservative extension does not change the original ontology as far as concepts built only from terms of the original ontology are used. The problem is to check whether a given extension is conservative wrt the original ontology. Very recently important research papers based on modal logics techniques have been published in important conferences on this topic, also by members of the Tones projects.

- *(to support error management)* Reasoning problems on finding explanation of consequences (respectively, non-consequences), and belief revision. In particular, the computation of Minimal Unsatisfiability Preserving Sub-TBoxes (MUPS) and the related reasoning task of axiom pinpointing.

Roughly, a MUPS for a concept is a minimal fragment of the ontology in which the concept is unsatisfiable. This is the basis to identify the precise position of errors in an ontology by, first, calculating MUPS and, secondly, maximally generalising ontology axioms.

Dually, as a non-standard ontology inference service, axiom pinpointing may provide a justification for any arbitrary entailment derived from an ontology. Given an ontology and any of its logical consequences, the axiom pinpointing service determines the premises in the ontology that are sufficient for the entailment to hold. In this context, the project aims to provide a formal notion of justification and propose a set of decision procedures for the axiom pinpointing problem.

- *(to support ontology bootstrapping)* The ontology designer wants to create an ontology about an enterprise domain that is already (partly) described in the form of a database schema. Instead of constructing the ontology from scratch, he wants automatic support for converting the database schema into an initial ontology. The ontology-based task required for this purpose derives an ontology from the database schema together with a set of views that connect the ontology with the database, so that queries over the ontology can be answered by using the data in the database. In the case of properly designed databases (e.g., in third normal form), this process may be deterministic and optimal. In the general case the process requires techniques similar to those of machine learning.

3.2 Information Access and Ontology Processing

If ontologies are used for problem-solving processes in applications, subproblems of high-level application tasks are formalised as decision problems over (one or more) ontologies. Hence, we speak of ontology-based tasks. Application programs will use ontology inference services to implement application services. Depending on the application problems, specific usage patterns for ontology services and communication architectures will emerge. Well-known standard (or basic) services such as satisfiability, subsumption, instance test, etc. will be used to realize high-level application services. Whereas these basic services might be sufficient for some ontology-based tasks in applications, an additional layer of inference services (also known as non-standard inference services) has been developed recently, and will allow for further formalisation of application subproblems. However, for the formalisation of even higher-level application-specific tasks (e.g., ontology-based information access, discovery and negotiation about services, etc.), new decision problems will have to be developed and implemented with ontology inference systems.

Implementations of non-standard inference services are often based on standard inference services, provided by current ontology reasoning systems. Application-specific tasks might depend on non-standard inference services or directly use basic inference services. Thus, from an abstract point of view we have a layer of inference services, and each layer accesses the services of one or more lower-level layers. The top-most layer is the application layer.

As usual in a standard decentralised computing scenario involving ontologies, the layered architecture can hardly be realized directly using different computational servers. The layered architecture is just a logical view. Indeed, it is important to find an appropriate allocation and distribution of computational processes and data such that efficient practical applications can be built. Ontology-based systems must support this in a methodological way. In the following, we therefore analyse several application tasks in which ontologies play a major role, and provide usage scenarios that determine how future systems will access and process ontologies in order to solve application problems in a distributed scenario. In particular, without claim of completeness, we discuss the following ontology-based tasks:

- Information access and intensional navigation
- Discovery and negotiation about services
- Media interpretation
- Configuration

We discuss these tasks in subsequent subsections.

3.2.1 Information Access and Intensional Navigation

An important goal in many applications is to *access information* with the mediation of an ontology, which models information domain constraints at the conceptual level. There are

mappings (in the simple case these are views or just one-to-one articulations) connecting the ontology vocabulary and the underlying information structures. The ontology defines a vocabulary which is richer than the logical schema of the underlying data, and it is meant to be closer to the user's (or the application's) rich (or diverse) vocabulary. The query is expressed using the ontology vocabulary, and it is evaluated, by means of reformulation or direct evaluation over the underlying information.

In addition, a *query support* task is meant to support a user in formulating a precise query – which best captures her/his information needs – even in the case of complete ignorance of the vocabulary of the underlying information system holding the data. The final purpose is to generate an ontology-based query ready to be executed by some evaluation engine associated to the information system. The intelligence of the interface is driven by the ontology describing the domain of the data in the information system. The user can exploit the vocabulary of the ontology to formulate the query, and she/he is guided by such a richer vocabulary in order to understand how to express her/his information needs more precisely, given the knowledge of the system.

Since the ontology expresses constraints over the data domain, the data can be possibly considered as dealing with incomplete information. The representation paradigm is therefore richer than in conventional database technologies. In fact, it is possible to refer to terms in the query which do not have a direct counterpart in the information source, but which can be materialised by means of reasoning over the ontology. In addition, it is possible to express the fact that the information about something is missing some data; again, reasoning at the level of the ontology may lead to some inference which completes the information.

The system allows to capture incomplete information in the domain, where the possible completions are constrained and computable by the ontology. Answers to queries will be given correctly and completely, by fully exploiting the ontology knowledge in order to fix the incomplete data. This technology – but with a less expressive kind of ontology language – is similar to the database technology for query answering under constraints.

The information system can be accessed using a richer vocabulary than the one used to structure the data. The user may adapt the ontology to his/her need or to the needs of the application connected to the information systems. Shared common ontologies may allow to access heterogeneous information sources in a uniform manner.

Two distinct approaches for implementation of ontology-based query answering are viable. The first one is based on Description Logics with ABox technologies, and it is geared towards expressive ontology languages, while, at the current state of the art, it does not scale well for very big datasets. This technology is derived from the classical Description Logic technology, which is amenable to many optimisations and special efficient cases. The second one is based on query reformulation on database technologies, and it is geared towards big datasets, while it does not allow for expressive ontology languages.

The problem of accessing information via ontologies becomes very relevant to convey semantic information during the query formulation. This allows users (or applications) to formulate queries also in the case of schema mismatch, since the information system is

accessed using the common and agreed-upon ontology vocabulary. So, the ontology plays the role of a vocabulary mediator between the data and the application/user.

Intensional navigation is the most innovative functional aspect of the query support task. Intensional navigation can help a less skilled user during the initial step of query formulation, thus overcoming problems related with the lack of schema comprehension and so enabling her/him to easily formulate meaningful queries. Queries can be specified through an iterative refinement process supported by the ontology through intensional navigation. The user may specify her/his request using generic terms, refine some terms of the query or introduce new terms, and iterate the process. Moreover, users may explore and discover general information about the domain without querying the information system, giving instead an explicit meaning to a query and to its subparts through classification. In the context of intensional navigation, the following are the added values of an ontology-based approach:

- Query validation. Incoherent queries - i.e., queries that can not return any value as answer, given their inconsistent meaning with respect to the schema - are detected before they are evaluated against the data.

- Query generalisation. In many situations, the query, even if it is consistent, can return an empty answer, since there is no actual document in the database satisfying it. In such cases, it is reasonable to generalise the query until a non-empty answer is obtained; the complete lattice of queries (w.r.t. containment) is the obvious space where such generalisations can be searched for.

- Query refinement. Queries can be specified through an iterative refinement process supported by the complete description lattice for the queries. This process is useful for data exploration tasks. The user may specify his/her request using generic terms; after the query classification, which makes explicit the meaning and the specificity of the query itself and of the terms composing the query, the user may refine some terms of the query or introduce new terms, and iterate the process.

- Intensional navigation. Users may explore and discover new generic facts without querying the whole document base, but by giving an explicit meaning to the queries through classification. The system has the ability of answering a query with synthetic concepts representing the general characteristics of the data that satisfy it, as opposed to answering with long sequences of detailed data. Moreover, if the query is classified in a taxonomy of descriptions and queries already computed and indexing the answers, then it can be processed with respect to the indexed objects only, rather than with respect to the whole data base. For this reason, an intensional answer can be considered complementary to the standard one, allowing a preliminary phase where the user can refine its query and exactly mark the boundary of what she/he is looking for.

Challenge
Answering queries using the vocabulary of an ontology mediating an information source.

Key benefits from an application point of view
Larger degree of autonomy of data from the application layer; unskilled users can access data in a more flexible manner.

Solution in terms of the application
Let user formulate queries at the ontology level (or at the level of conceptual data modelling) rather than at the implementation level (or logical level). Give support for query formulation and reformulation.

Added value by using ontologies
Dealing with incomplete information; query answering with reasoning under the ontology constraints; support for sensible ad hoc queries due to ontology-based suggestions for query reformulations.

Technology for implementing the solution
Efficient ABox systems, expressive query languages, query reformulation systems over SQL or other DB technologies.

3.2.2 Discovery and Negotiation about Services

Recently, computational resources for application subtasks are made available using standard network technology as so-called *services*. While in early approaches services are described by *names* that are stored in a repository (probably using hierarchical structures to classify services), recently, the necessity for *semantically deep descriptions for services* has been recognised in order to be able to build flexible architectures and loosely coupled systems. Given a semantics-based description of services (supplies) and requirements (demands) a formal definition for the suitability of a supply for a certain demand has to be found. In the literature, several approaches for a "match" of a supply with a demand are discussed [8, 14, 6]:

- Satisfiability of the conjunction of supply and demand description, i.e., there is one "world" (model of the background ontology) that satisfies supply and demand in combination (usually this is a very weak test).

- Subsumption between supply and demand descriptions w.r.t. a background ontology, i.e., (i) if the *supply is the subsumer*, it is more general, and hence provides even more than required, or (ii) if the *supply is the subsumee*, it is a "plugin" (or component) for what is required (usually both definitions result in very strong tests).

- Entailment of non-disjointness, i.e, is it the case that in all models of the ontology the conjunction of supply and demand descriptions is satisfied.

However, in all cases, it is not sufficient to just find out that a match is possible. Indeed, in the first and third cases, it would be interesting to retrieve the model(s) and further process it to actually implement a concrete service call in the application. In the second case, an explanation about why a certain subsumption relation holds might lead to a concrete implementation of a service call.

In general, the formal definition of matching is application-dependent. While the latter tests seem promising in some applications, it is obvious that in case a non-match of a demand description with all supply descriptions in a repository is found, on the one hand one is interested in what is missing by the supplies and, on the other hand, what requirements should be dropped from the demand. The former can be formalised as abduction whereas the latter can be expressed as a specific form of contraction. For instance, if subsumption (supply to be more general than the demand) is used as a formalisation of matching a supply description σ with a requirement description ρ, the task is to compute concepts δ and κ such that $\rho \sqcap \kappa \sqsubseteq \sigma \sqcup \delta$.

Given the possibility to compute abductions and contractions for matchmaking, a formalised model for a negotiation process can be developed. Note that service descriptions might involve application-specific data as well (e.g., taken from an enterprise resource planning system).

Challenge
Find high-level application services based on declarative descriptions and with reference to ontologies. Adapt requirements (demands) to available services (supplies) in order to allow for service negotiation.

Key benefits from an application point of view
Semantics-based service discovery provides for larger flexibility than simple name-based service discovery and invocation.

Solution
Formalisation of discovery and negotiation as matching supplies against demands using ontology-based inference services.

Added value by using ontologies
Expressive descriptions for domain knowledge provide for semantics-based service characterisations.

Technology
Formalisation as satisfiability of conjunction of demand and supply descriptions (overlap in one model), as subsumption (either supply is more general or demand is more general, depending on the application context), or formalisation of matching as entailment of non-disjointness (overlap in all models). Ability to identify missing things in supplies (via abduction) or determine over-constraining requirements (via contraction).

3.2.3 Media Interpretation

For the interpretation of different media with modalities such as natural language, images or image sequences (and possibly combinations of different modalities) it is well-known that extensive background knowledge provides the basis for a deep-level understanding. Ontologies based on logics provide the backbone of such knowledge in practical approaches, and logical formalisation of media interpretation at a deep level have been proposed in the literature.

Given the background knowledge Σ and an appropriate representation of what is perceived from media using some formulae Φ, several different formalisations have been investigated.

In explanation-based approaches, the challenge is to compute some formulae Δ such that the percept Φ can be "explained" w.r.t. the background knowledge Σ: $\Sigma \cup \Delta \models \Phi$. Hence, media interpretation is formalised as abduction [11]. Since descriptions of percepts usually also have the status of assumptions due to ambiguities at the signal level, some perceptual descriptions Ψ might possibly be retracted. Thus, in general, the challenge is to compute Δ and Ψ such that $\Sigma \cup \Delta \models \Phi \cap \Psi$ with Δ and Ψ being minimal according to some application-specific norm.

However, not all aspects of media interpretation are naturally seen as explanations. Therefore, in other approaches, experience-based completions of high-level descriptions are used to formalise media interpretation in terms of configuration [7] (see also below). In formal terms, a model $\mathcal{I}$ is computed such that $\mathcal{I} \models \Sigma \cup (\Phi \cap \Psi)$ (with Ψ being minimised). Note that interpretation results can be manifold and multiple models $\mathcal{I}$ might be possible. The common core of the models representing interpretation results can be represented as an ABox.

Challenge
Provide high-level propositional descriptions of media content.

Key benefits from an application point of view
Media interpretation can help in semantics-based information access to media content repositories, and ontologies play a major role in this process.

Solution
Representation of interpretation tasks as logical decision problems w.r.t. ontologies providing the background knowledge.

Added value by using ontologies
Ontologies allow for a formal representation of background knowledge, a logic-based formalisation of interpretation tasks allow for the evaluation of interpretation algorithms and, therefore, has advantages over ad hoc solutions.

Technology
Abduction, contraction, model computation.

3.2.4 Configuration

Ontologies have been successfully used to implement configuration systems (see e.g. [9]). The idea is to use an ontology language to describe the configuration space and an initial configuration to be automatically completed such that some constraints are met. However, early approaches were based on inexpressive ontology languages, i.e., many constraints could not be expressed in a declarative way, and a lot of programming was still required.

With expressive ontology languages, domain-specific constraints can be declaratively expressed and with expressive query languages, many properties of the final configuration can be queried, maybe resulting in further constraints being declared.

A usual formalisation of ontology-based configuration tasks is to define the final configuration as a model of a description of the initial configuration (ABox) w.r.t. to a knowledge base (TBox).

Challenge
Ontology-based configuration with access to application-specific databases describing information about certain parts to be configured.

Key benefits from an application point of view
Declarative specification of the configuration space provides for flexibility if requirements change.

Solution
Formalisation of ontology-based configuration as a formal decision problem.

Added value by using ontologies
Descriptions of the configuration space can be evaluated during the design phase. Domain-specific vocabulary can be captured.

Technology
Compute a model and further process it with respect to application-specific requirements.

3.2.5 Architectural Implications for Ontology Access and Processing

With reference to ontology access and processing some research goals become apparent that apply to all use cases mentioned in previous sections in order to avoid communication

overhead in decentralised computing architectures. We can extend the set of server-side capabilities of ontology processing systems, or allow for efficient client-side implementation of application-specific reasoning services. If server-side extensions are considered, application-specific data must be made available to ontology processing systems whereas if client-side extensions are considered, efficient access to ontology information is required to allow for scalable systems. Both view are complementary, and in both views non-trivial software-engineering problems must be solved:

- Client-side caching architectures to avoid repetitive queries and corresponding result sets to be sent over the network.
- Definition of expressive TBox and ABox query languages (move the queries not the data/data descriptions).
- Support for application-specific server-side extensions with, for instance, logic programming techniques in order to avoid frequent calls to server-side inference tasks such as subsumption tests or ABox satisfiability tests from a remote client application (move the algorithms not the data/data descriptions). This also is particularly important if operations on computed models are to be performed since models tend to be very large (and can be infinite such that model descriptions have to be processed).
- Interpretation of data in databases as "virtual" ABoxes, i.e., data descriptions w.r.t. an ontology that are loaded "incrementally and on demand" into ontology processing systems.

3.2.6 Ontology-based Research Tasks

The ontology-based research tasks that are at the core of the abovementioned technologies for the information access and ontology processing activities are summarised in this Section.

- *(to support information access)*

  Virtual ABoxes whose assertions are derived from database contents, persistent ABoxes, storage hierarchy for ABoxes with caching and loading on demand in order to cope with extended expressivity. Loading on demand can benefit from computing abstractions or approximations of ontologies and retrieving data into ABoxes using transformation-based approaches

- *(to support discovery and negotiation about services)* Preference or utility-based abduction and contraction. Ontology-based matching operations that refer to actual data of a concrete supply (e.g., provided by enterprise resource managing systems: Is a required part required for a particular service invocation actually available on stock within a reasonable amount of time?). Formalisation of matching as satisfiability of conjunction of demand and supply descriptions can benefit from actually inspecting

a certain model (the overlap). Model generation has not been investigated in depth in description logic systems.

- *(to support media interpretation)* Preference or utility-based abduction and contraction. Computation of preferred models based on probabilities.
- *(to support configuration)* Optimisation techniques for combining ontologies and logic programming (description logic programs with answer-set or well-founded semantics) used to generate configurations. Optimisations for probabilistic description logic programs to generate preferred configurations.

3.3 Ontology Integration and Segmentation

Challenge To develop services and algorithms for ontology integration and segmentation.

Key benefits
Re-use of existing ontologies in a multiple ontology environment, both through integration as well as modularisation. Support for unskilled users.

Solution
Design and implementation of new reasoning-based services specific for ontology integration and segmentation. Development of suitable end user support in ontology development tools.

Added value
Ability to predict and formally understand the properties of the integrated or segmented ontology. Development of generally applicable integration and segmentation techniques.

Technology
Non-standard reasoning tasks.

Ontology-based Research Tasks
Suitable non-standard reasoning tasks and algorithms for ontology integration and segmentation. Clarification of the relationship between integration and segmentation. Development of suitable user interfaces for these tasks in ontology editors.

3.3.1 Challenge

The acceptance of the Web Ontology Language (OWL) as a standard will facilitate the proliferation of independently developed ontologies. In this scenario, different ontologies need to be confronted and related to each other to produce a single integrated and reconciled ontology that deals with a larger domain of interest. Often, the ontologies to be

integrated have been developed by different groups of experts, which makes the integration process a very difficult task. So far, ontology integration problems have been mostly tackled in an ad-hoc manner, with little or no tool support and with no clear notion of what to expect from the integration. In particular, it is possible that unintended consequences are introduced, even if the ontologies to be integrated are widely tested and understood, and such consequences can be far from obvious. There appears to be consensus that appropriate support for ontology integration is not provided by current ontology editors and browsers, nor is the problem sufficiently understood at a more fundamental level. The need for extensive support for ontology integration tasks reasoning services has been widely acknowledged by the Ontology Engineering and Semantic Web communities.

Complementary to the problem of Ontology Integration, and equally important, is the problem of Ontology Segmentation, often referred in the literature as Ontology Modularisation and Ontology Partitioning. In Ontology Engineering, as in Software Engineering, *modularity* is a much praised virtue. Modular representations (or programs) are easier to understand, verify, debug, extend, reuse parts of, and thus facilitate collaborative development. For Web ontologies, where the collaboration is, in large part, uncoordinated, it is often not enough that the ontology be modular in a general sense, but that, for a large ontology, there are extractable parts that can be reused outside the context of the original ontology. Furthermore, there is the expectation that those fragments are not *arbitrary*, but maintain some relation to the meaning of those parts in the original context. Indeed, if the fragments are "modules", one would expect that their extraction preserves key aspects of their embedded meaning. Basic to a clear notion of modular decomposition of a logical theory (such as an ontology) is an account of the the *correctness* of that decomposition and a clear notion of the consequences of the extraction of modules from its original context. However, the problems of *formally* characterising a modular representation, on the one hand, and of automatically identifying modules within an ontology, on the other, have not been satisfactorily addressed in the Ontology Engineering and Semantic Web literature and remain widely unexplored, although their relevance has been widely accepted by those communities.

3.3.2 Key Benefits

In order to overcome these problems, ontology engineers need a clear notion of what to expect from both the ontology integration and ontology segmentation processes. Without such a fundamental understanding of the critical issues involved in these problems, the ontology engineer is at a loss when performing these tasks. The result is the adoption of ad-hoc and highly unpredictable techniques as a common practice, which often leads to undesired results. These techniques are sometimes developed for specific ontologies and thus are not generally applicable.

The development of suitable, well-understood, services for ontology integration and segmentation is key for collaborative development of ontologies, which is expected to be the most common situation in Semantic Web applications. In particular, the following application scenarios, among others, would benefit from the availability of these services:

- A set of ontologies that have been developed independently and now are required to inter-operate in an application. This scenario comprises, for example, the following situations:
  - Different departments within an organisation, say an aircraft industry, have developed their own ontologies. In particular, three different departments might have developed respectively an ontology about engines, aircrafts and employees.
  - An online travel booking company wants to reuse ontologies about hotels, flights, destinations and vehicle rentals developed by other organisations and probably available somewhere on the Web and relate them by adding new knowledge.
- An organisation wants to build an ontology in order to provide meaning to its data. The ontology to be designed contains information about different application domains. For example, a company might be interested in building an OWL ontology that models its organisational structure, which should contain knowledge about employees, government organisations, funding resources, projects, etc. In such cases, it is likely that the modelling of the different application domains will be assigned to different groups of experts, which would only need to communicate whenever a change performed by some group in its ontology is affecting the knowledge in another ontology, under the responsibility of a different group.
- An organisation has developed a large ontology and wants to assign to different engineers the maintenance and evolution of different parts of it. In order to save time, effort and money in such a process, it is required that the segmentation process is performed in a well-understood way.
- A physician wants to annotate some clinical data using the name of some drug, has searched the Web and found two ontologies that represent such a drug. The physician needs to decide which ontology to use. For such a purpose he/she wants to understand the meaning of the drug in both ontologies, without having to understand both ontologies as a *whole*. In order to accomplish this task, the availability of a segmentation algorithm that allows to retrieve only the fragments that are relevant to the meaning of a certain concept in a given ontology is a must.

The support for ontology integration and segmentation tasks in ontology design tools is also critical. The addition of new services to an ontology editor requires addressing important issues, such as the design of suitable user interfaces, to assist the ontology modeller.

3.3.3 Solution

When integrating well-established and tested ontologies, it is of prime importance for the ontology developer to have control of the consequences of her modifications and to have

formal guarantees that, after the integration has been performed, no unwanted consequences will be obtained. What exactly to consider as an unexpected result may vary depending on the specific role that each of the input ontologies is playing in the integration.

Two basic ways of integrating ontologies seem to have become a standard in the Ontology Engineering community:

1. An ontology (or set of ontologies) dealing with a certain subject matter are integrated with an *upper ontology*, that describes more general terms, which are often domain-independent. Several well-known ontologies have already been developed specifically to be used as formal upper ontologies. Prominent examples are the Suggested Upper Merged Ontology (SUMO) and DOLCE. The IEEE Standard Upper Ontology Working Group is in the process of developing a standard upper ontology to be used in a wide variety of applications.

2. Different ontologies describing largely different subject matters are integrated to describe a broader subject. The maintenance and evolution of each ontology can be then assigned to a different group of experts. This is the case of some prominent ontologies such as the OWL-S ontologies, NASA's SWEET Ontologies and the National Cancer Institute (NCI) Thesaurus.

For each of these scenarios, different properties of the integrated ontology may be expected. For example, when merging a domain ontology with its upper ontology, we may well expect that: **1)** the merged ontology will be consistent; **2)** no new subsumption relations will be inferred between the concepts originally contained in the domain ontology; **3)** no new subsumption relations will be inferred between the concepts in the upper ontology; **4)** new subsumption relations will be inferred between the concepts in the domain ontology and the concepts in the upper ontology. However, in the case of merging ontologies describing largely different subject matters, we would not expect new subsumption relations to be inferred between concepts originally contained in different ontologies.

Analogously, for the ontology segmentation problem, different properties of the retrieved fragments may be imposed. We believe that a sensible approach to the ontology integration and segmentation problems should take into consideration the specificities of each integration paradigm.

3.3.4 Added Value

So far, ontology integration and segmentation problems have been mostly tackled in an ad-hoc manner, with little or no tool support and with no clear notion of what to expect from the integrated or segmented ontology. The result is the adoption of ad-hoc and highly unpredictable techniques as a common practice, which often leads to undesired results. These techniques are sometimes developed for specific ontologies and thus are not generally applicable. Our approach to the integration and segmentation tasks aims at

acquiring a formal, in depth, understanding of the problems; the goal of our techniques is to predict and understand the properties of the integrated or segmented ontologies and to develop generally applicable integration and segmentation techniques.

3.3.5 Technology

The proposed solution is based on the availability of novel non-standard ontology-based reasoning tasks. Some of these tasks, such as deciding conservative extensions in the case of ontology integration and uniform interpolation in the case of ontology segmentation, go beyond standard reasoning techniques. It is our goal to develop practical reasoning algorithms for these tasks and integrate them in state of the art DL reasoners and integrated in ontology design tools.

3.3.6 Ontology-based Research Tasks

The ontology-based research tasks that are at the core of the abovementioned technologies for ontology integration and segmentation activities are summarised in this Section. The unifying factor of this research area is ontology *modularity* in its aspects of ontology integration, ontology segmentation and reuse, and ontology change. The research aims:

- to identify and systematise the most prominent paradigms for ontology integration;
- to develop services and algorithms for assessing the impact of integrating a set of ontologies into a single, reconciled ontology;
- to develop services and algorithms for segmenting ontologies into meaningful parts and to understand and predict the relationship between the segments and the parent ontology;
- to understand the implications of the different integration paradigms on the intended and unexpected consequences;
- to gain an understanding of what makes ontologies suitable for integration, thus enabling the definition of ontology *modules*;
- to understand and formalise the relationship between the ontology integration and ontology segmentation problems;
- to develop suitable end user tool support for ontology integration and segmentation;
- to understand the effect of the expressivity of the ontology language on the ontology integration and segmentation tasks.

3.4 Ontology-mediated Interoperation

Challenge
Interoperation between a conceptual layer constituted by an ontology and a data layer constituted by a traditional data management system; interoperation among different ontology-based peers.

Key benefits
Peer-to-peer data integration. Semantic Web.

Solution
Query Answering; mapping specification; mapping verification; query reformulation; instance exchange; inconsistency management; update at the instance level; autoepistemic reasoning over ontologies; epistemic reasoning over multiple ontologies

---

Added value
The mediating ontology provides a unified view through which the local information can be accessed and integrated.

Technology
LAV and GAV mappings, databases with incomplete information, reasoning with constraints.

3.4.1 Challenge

In this section we consider the problem of interoperability with information sources mediated by ontologies. This is the problem of combining data residing at autonomous, heterogeneous sources, and providing the client with a unified, reconciled global view of these data. The user queries the global schema, ignoring the location and structure of the data sources, and leaving to the system the task of merging and reconciling the data at the sources. Typically, sources adopt different ontologies, models, and systems for storing data. The use of a conceptual data model has been advocated for providing an appropriate abstraction of all the data residing at the sources. We consider two main scenarios. In the first scenario we have an ontology that is connected with a traditional kind of information system, e.g. a DBMS, external to the ontology. In this setting the ontology

constitutes a *conceptual layer* which represents the conceptualisation of the domain of interest, whereas the DBMS constitutes a *data layer* containing data related to the domain of interest. Since these two subsystems are independent one another, explicit mappings establish the connection between the two systems. In particular, through these mappings data are extracted from the data layer and cast into information at the conceptual layer. Notice that the mapping has to resolve the dichotomy existing between value-based representations typical of the relational technology currently used in DBMSs and object-based representations typical of ontologies, which are currently realized through class-based formalisms, such as Description Logics. Notice that the above setting can be immediately extended to consider several independent DBMSs constituting the data layer. In this sense, such a scenario generalises to the presence of ontologies typical data integration scenarios studied in the database literature.

In the second scenario, ontology-based peer-to-peer interoperation is the problem of allowing various ontologies to interoperate and exchange information, without necessarily providing a global ontology as the principle mean of accessing information in them. The design of an ontology-based peer-to-peer integration system is a very complex task, which comprises several different issues: (i) dealing with heterogeneity of the peers, i.e., typically peers adopt different ontologies, models, and systems for storing data; (ii) specifying the mapping between the peers; (iii) processing queries expressed on a peer on the basis of the knowledge and the data managed at each interoperating peer.

3.4.2 Key Benefits

With respect to the first scenario, many organisations face the problem of integrating conceptual schemas or ontologies modelling information systems in several sources. Companies that build a Data Warehouse, a Data Mining, or an Enterprise Resource Planning system may take advantage of the the conceptual level description provided by the ontologies, abstracting from how data are logically organised where they are stored.

The second scenario is instead typical of the semantic web, where independent ontology are produced and relationships among them are established to allow them to interoperate and exchange information. Also in a smaller scale peer-to-peer interoperation is of great interest, e.g., when few independent organisations decide to cooperate allowing a certain amount of their information to be exchanged still maintaining an autonomous management of their own information.

The web is constituted by a variety of information sources, each expressed over a certain ontology, and in order to extract information from such sources, their semantic integration and reconciliation in terms of a mediated ontology is required.

3.4.3 Solution

The main tasks of the above scenarios are:

- *Query Answering.* The user poses a query to the ontology she/he/it is accessing. The answer to this query is generated activating the mappings towards the external

sources (the data layer in the first scenario, the ontologies in the other peers in the second scenario).

- *Mapping Specification.* Specification of the semantic relationship between the conceptual and the data layer in the first scenario and the conceptual layers of the different peers in the second scenario.
- *Mapping Verification.* Analysis and validation of the mappings described above.
- *Query Reformulation.* Given a query posed over an ontology, construct the queries over the external sources (data layer in the first scenario, the ontologies of the other peers in the second scenario) whose answers contribute to answer the initial query.
- *Instance Exchange.* Materialisation of instances of the ontology that capture the information extracted from the external sources.
- *Inconsistency Management.* Since we are talking about independent systems, inconsistencies on the information represented in them may naturally arise. Management of such inconsistencies is crucial for allowing seamless interoperation between conceptual and data layer in the first scenario and among different peers in the second scenario.
- *Update at the instance level.* This is the task of handling the situations in which extensional information changes. Updates may be propagated from the conceptual layer to the data layer in the first scenario and from the ontology of one peer to the ontologies of the interoperating peers in the second scenario.
- *Autoepistemic reasoning over ontologies* (aka introspective reasoning). This is the task of reasoning on the state of the ontology rather than reasoning on the state of the world represented by the ontology. This is a very important form of reasoning for meta-level analysis of ontologies.
- *Epistemic reasoning over multiple ontologies.* This is specifically of interest in the second scenario when one ontology is interested in modelling the state of the other ontologies, i.e., what the ontology knows about the other ontologies.

3.4.4 Added Value

The interoperability problem is one of the basic problems in the development of techniques for the semantic web. Indeed, the web is constituted by a variety of information sources, and in order to extract information from such sources, their semantic integration and reconciliation is required. In this context we have various local ontologies, developed independently from each other, and we are required to build an integrated, global ontology as a mean for extracting information from the local ones. Thus, the main purpose of the global ontology is to provide a unified view through which we can query the various local ontologies.

3.4.5 Technology

The problem of designing effective interoperation systems has been addressed by several research and development projects in the last years. As noted above, interoperation systems are based on a unified view of data, called mediated or global ontology, and on a module, called mediator, that collects and combines data extracted from the sources, according to the structure of the mediated schema. A crucial aspect in the design and the realization of mediators is the specification of the relation between the sources and the mediated schema. Two basic approaches have been proposed in the literature, called global-as-view (or simply GAV) and local-as-view (or simply LAV). In the GAV approach, a view over the sources is associated to each element of the global schema, de- scribing how to populate such an element using the data at the sources. Most data integration systems adopt the GAV approach.

The method for computing the answer to queries posed in terms of a global schema in an interoperation system is typically based on query reformulation. For this purpose, the system should be able to re-express the query in terms of a suitable set of queries posed to the sources. In this reformulation process, the crucial step is deciding how to decompose the query on the global schema into a set of subqueries on the sources, based on the meaning of the mapping. The computed subqueries are then shipped to the sources, and the results are assembled into the final answer. It is well known that processing queries in the local-as-view (and global-as-view as well) approach is a difficult task. Indeed, in this approach the only knowledge we have about the data in the global schema, together with the ontology, is through the views representing the sources, and such views provide only partial information about the data. Therefore, extracting information from the system is similar to query answering with incomplete information, which is a complex task. This framework poses new challenges, specially related to the need of taking the semantics of the conceptual global schema into account during query processing. The idea of adopting a conceptual data model for expressing the global schema makes query processing more involved than in the simplified framework usually considered in the literature. As a matter of fact, the semantics of a data integration system is best described in terms of a set of databases, rather than a single one, and this implies that query processing is intimately connected to the notion of querying incomplete databases.

3.4.6 Ontology-based Research Tasks

The ontology-based research tasks that are at the core of the abovementioned technologies for ontology-mediated interoperation activities are summarised in this Section. In the last Chapter an example will be fully worked out.

Reasoning problems motivated by this task:

- *Ontology-to-data storage.*
  The simplest setting is the one where we have one relational table for each concept and for each role: direct mapping. Already in this simple setting, several challenges arise:

  - "Impedance mismatch" between objects in the ontology and values in the database
  - Query and update reformulation: queries and updates posed in terms of the ontology should be reformulated in terms of queries posed to the database
  - If the database is large, computation going beyond SQL (Log-space data complexity) might be problematic: integration with light ontologies

- *Ontology-to-data sources.*
  The research challenges in this area are:
  - How to express mappings
  - "Impedance mismatch" between objects in the ontology and values in the database
  - Instance extraction, cleaning and reconciliation
  - How to deal with inconsistencies between the data and the ontology
  - How to (correctly and efficiently) answer queries expressed on the ontology
  - How to process updates expressed in terms of the ontology

- *Ontology-to-ontology (classical)*
  - Mappings are again crucial
  - How to exchange instances
  - How to reformulate queries
  - How to deal with inconsistencies

- *Ontology-to-ontology (P2P)*
  - (Distributed) Query processing
  - Data exchange
  - Inconsistencies
  - Quality (e.g., trust)
  - Privacy

4 Four Concrete Example User Scenarios

In this final section we will show few basic common use cases, emphasising in particular the added value of the support of ontology-based tasks. These use cases adopt as a neutral language to represent ontologies UML-like class diagrams. The advantage of this choice is that examples may be clear even to people who do not have familiarity with more classical ontology languages, due to its graphical nature and to the fact that UML class diagrams are quite widespread within computer technologies. Note that we use the notation of cardinality constraints as *participation constraints* as opposed to the UML standard that uses them as *look-through constraints*.

4.1 Ontology Design

Complete reasoning over expressive ontology languages supports the ontology engineer in the creating and maintaining ontologies tasks. The following examples show the kind of insights that a reasoning enabled system could provide during the ontology editing.
Let us consider the ontology represented by the diagram in Figure 6. The ontology states that mobile calls are a kind of calls (IsA link between entities); that phone points are partitioned between cell points and landline points (i.e., any phone point is either a cell or a landline point, but not both: they form a covering and disjoint IsA hierarchy). Each call has at least one destination phone point (mandatory participation of `cell` to `destination`), while it has exactly one origin phone point. Mobile calls are related to cells through the `mOrigin` relationship. Finally, the binary relationship `mOrigin` is a included in the binary relationship `origin`.

Which are the consequences of the above ontology? The system could automatically complete the diagram in the way depicted in Figure 7. The added constraint states that in the above scenario it is necessarily true that each mobile call may have an origin from at most one cell point. In order to understand why this is true, consider the following. The `mOrigin` binary relationship is included in the `origin` binary relationship, i.e., any pair in `mOrigin` is also among the pairs in origin. Since each call participates exactly once as first argument to the `origin` relationship, if I take a generic sub class of calls,

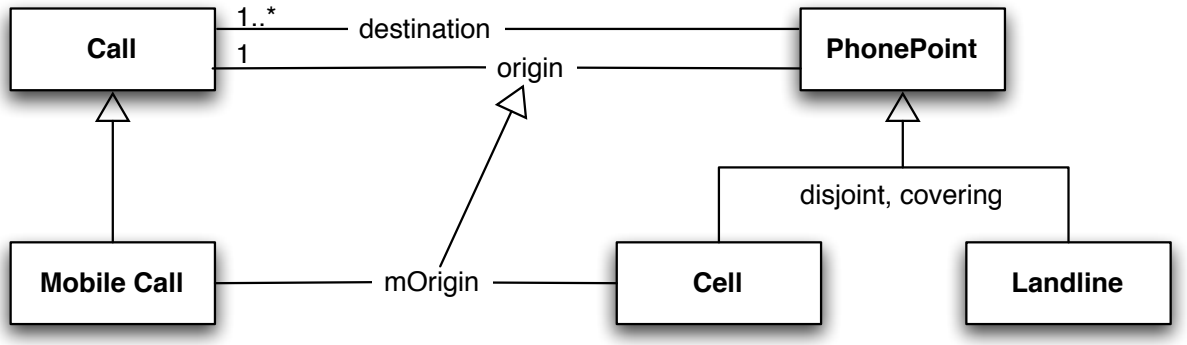

Figure 6: First ontology design scenario.

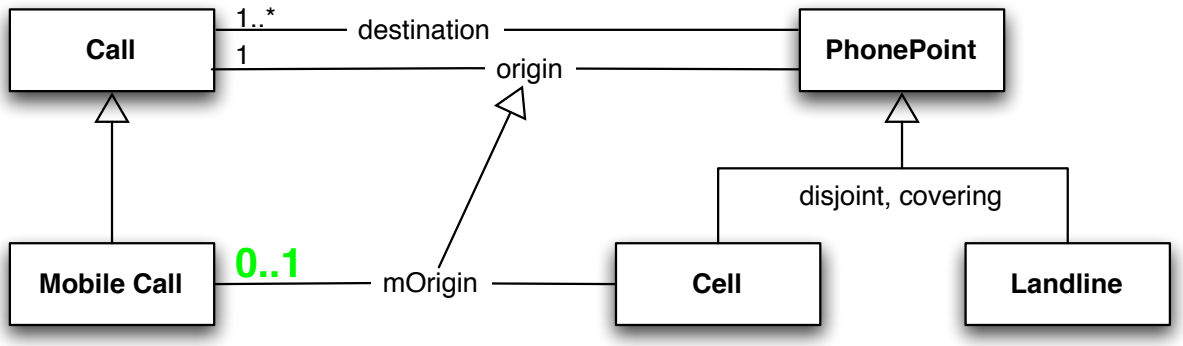

Figure 7: First ontology design scenario with deductions.

such as the class of mobile calls, and a sub relationship of the `origin` relationship, such as `mOrigin`, then we can conclude that necessarily each mobile call participates at most once as first argument to the `mOrigin` relationship. Nothing can be concluded about the minimum participation, since the `mOrigin` relationship may not contain all calls in the `origin` relationship.

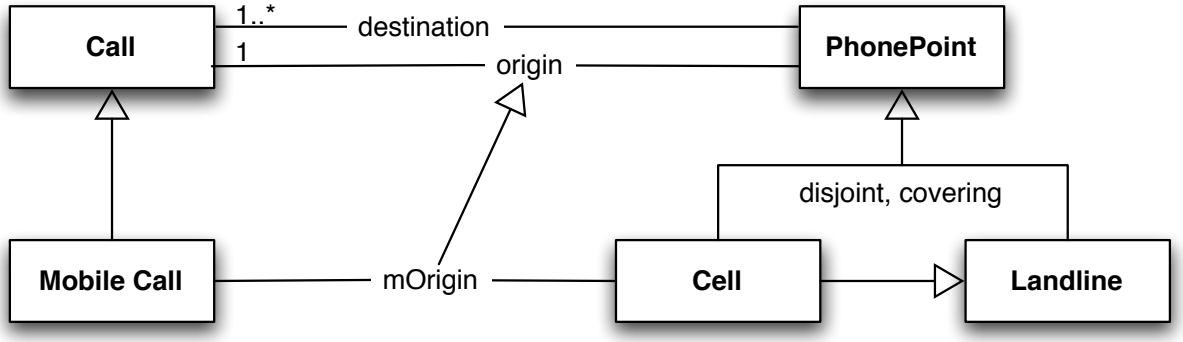

Figure 8: Second ontology design scenario.

Let us now consider the same diagram as Figure 6 with an additional IsA link stating that each cell point is among the landline points (see Figure 8). The system is now able to conclude the following new facts (see Figure 9). The `cell` entity is inconsistent, i.e.,

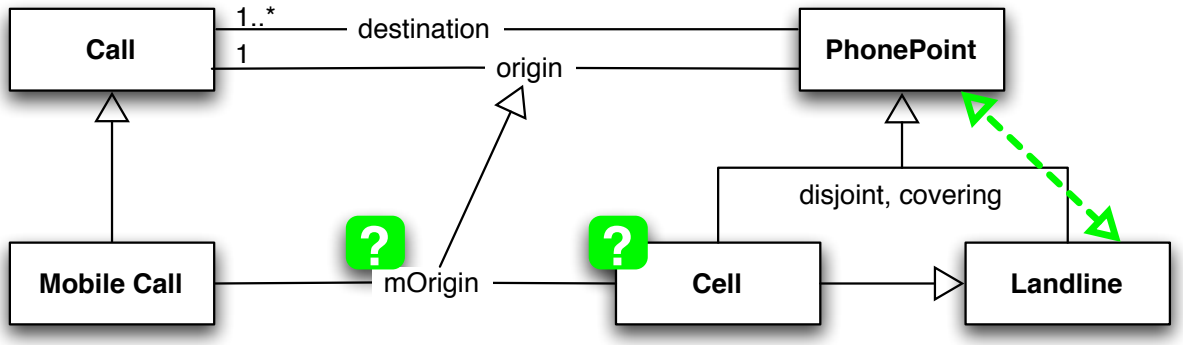

Figure 9: Second ontology design scenario with deductions.

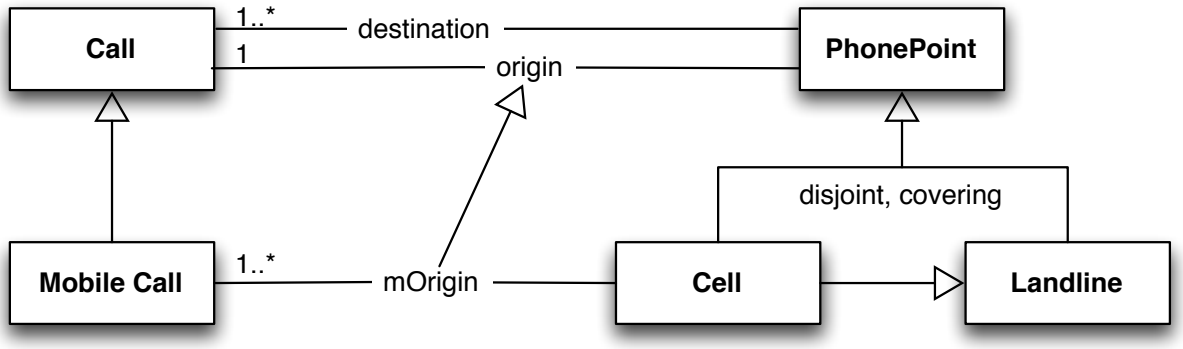

Figure 10: Third ontology design scenario.

does not have any instance, since the disjointness constraint in the IsA link states that there is no element in common between cell and landline phone points. The empty set denoted by the `cell` entity is the only set which can be at the same time disjoint and a subset of another set. Since the phone point entity is formed by the union of the cell and landline entities, and the cell entity is inconsistent, the landline entity becomes equivalent to the phone point entity. Moreover, since there is no cell point, there is no pair in the `mOrigin` relationship as well (i.e., it is inconsistent): the diagram states that any second argument of the `mOrigin` relationship should be of the cell type. The `Mobile Call` entity is not inconsistent, since it may be populated by calls which have no `mOrigin` at all (this is possible, since there is no mandatory participation constraint).

Let us now add a cardinality constraint, stating that now each mobile call should participate at least once to the `mOrigin` relationship (i.e., a mandatory participation constraint). The change results in the diagram of Figure 10. Now the system deduces that the mobile call entity is inconsistent as well (see Figure 11).

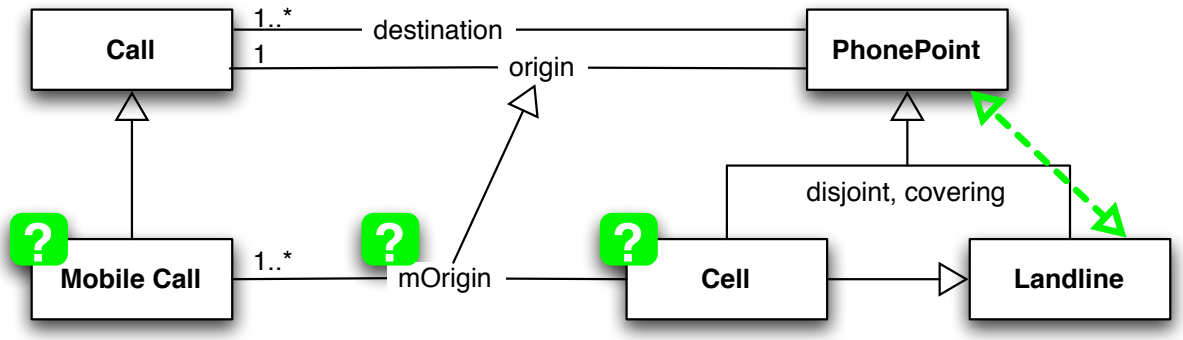

Figure 11: Third ontology design scenario with deductions.

4.2 Information Access

In the task of query answering over data sources mediated by ontologies, Complete reasoning is essential to obtain complete answers. The reason is that, in general, the actual data sources are *incomplete* w.r.t. the ontology describing them. To show that, even in

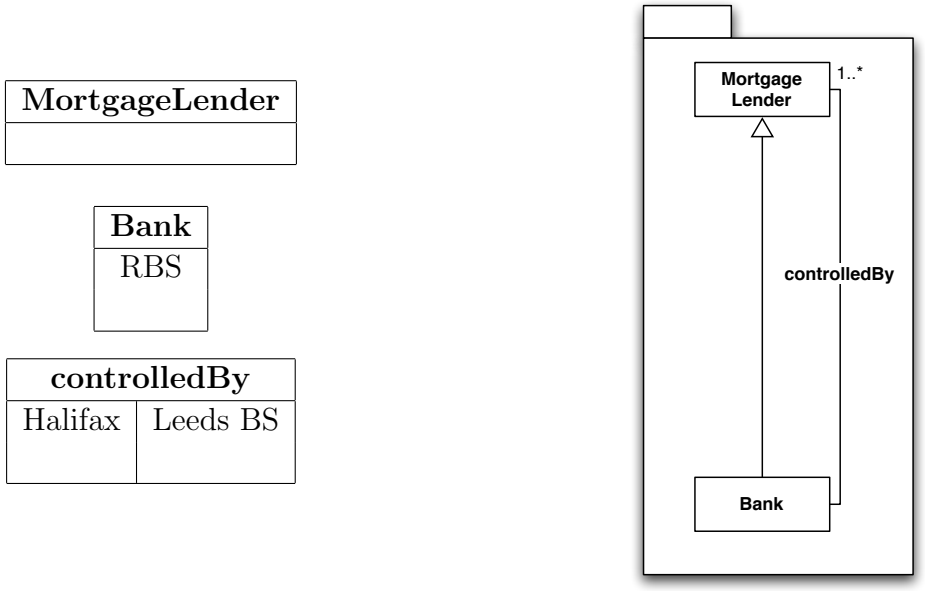

Figure 12: Information access scenario.

simple cases, the query answering process is not trivial and requires some sort of automated reasoning, we consider the right side ontology of Figure 12. Moreover, we assume that the binary relationship `controlledBy`, relates mortgage lenders to banks (i.e. we fix the first and second arguments).

To the sake of clarity, we assume that for each class and binary relationship there is a corresponding unary and binary table in the underlying database. Note that these can be actual tables or views over the real data. In this example the logical schema of the data source contains two unary tables – $\texttt{mortgage\_lender}_T$ and $\texttt{bank}_T$ – and the binary table $\texttt{controlled\_by}_T$.

Let us consider a database where the table $\texttt{mortgage\_lender}_T$ is empty, $\texttt{bank}_T$ contains a single element "RBS", and $\texttt{controlled\_by}_T$ a single pair ⟨ "Halifax", "Leeds BS" ⟩. Where "RBS", "Halifax", "Leeds BS" are distinct constants. This correspond to the tables in Figure 12, which are sound views over the source data.

Consider a query to retrieve the projection over the first argument of the relationship `controlledBy`; i.e.:

$$Q(x) \leftarrow \texttt{controlledBy}(x, y).$$

Simply retrieving the content of the corresponding table ($\texttt{controlled\_by}_T$) is not enough. In particular, given the database it would return just Halifax. In fact, a complete answer should contain all the elements "RBS", "Halifax", and "Leeds BS".

To understand the reasons for "RBS" and "Leeds BS" being part of the answer, consider Figure 13 (in *italic* the inferred data); we should consider that the class `Mortgage Lender` has a mandatory constraint on the relationship `controlledBy`. Therefore every element of `Mortgage Lender` is necessarily participating as first argument of the relationship. The actual $\texttt{mortgage\_lender}_T$ is empty, but every element of class `Bank` is a member of the `Mortgage Lender` class; therefore, "RBS" should be part of the answer.

MortgageLender
RBS
Leeds BS

Bank
RBS
Leeds BS

controlledBy	
Halifax	Leeds BS
RBS	...
Leeds BS	...

Figure 13: The tables with the virtual data.

Moreover, "Leeds BS" is participating in the `controlledBy` relationship; therefore "Leeds BS" is an element of class `Bank` as well (because of the range restriction).

4.3 Ontology Integration

This example shows that ontology reasoning can provide essential support in the ontology integration process, by exposing serious but subtle conceptual mistakes. The task we analyse consists in integrating two given ontologies.

Consider the two ontology integration example depicted in Figure 14. Each ontology could have been designed for a specific purpose, so they not necessarily share the same assumptions about their domains.

The left side ontology describes lending institutions, which are partitioned into public lenders and building societies. Moreover, the ontology asserts that Public lenders are no profit companies, and that banks are not building societies.

In the right side ontology, we have the generic class of mortgage lender institutions which specialises into the bank class. We also assume that actually the right ontology describes a world which is actually a portion of the world described more accurately by the left ontology.

A very natural integration between the two ontologies is pursued by the ontology engineer: she/he states that banks of the lower ontology are among the banks of the top ontology, and that lending institutions of the lower ontology are fair lenders and profit companies as defined in the top ontology. The integration is shown in the figure by means of thick inclusion statements.

The consequences of this integration attempt are depicted in Figure 15. It turns out—from the big question mark at the corner of the `Bank` class—that no banks can exist according to the integration of the two ontologies! This is somewhat unexpected, since we thought we were playing a rather simple game in this case. Why is this? The reason is the following. First of all, we can derive that lending institutions are building

society (as pointed out by the tool); in fact, mortgage lenders are lending institutions, which can be either public lenders of building society. On the other hand we have that lending institution are profit companies, which are provably disjoint from public lenders. therefore, any mortgage lender has necessarily to be a building society. At this point, we are closer to the answer to our original question about the inconsistency of the `Bank` class. Those right side ontology banks are at the same time a kind of top ontology banks and building societies (by transitivity). However the two latter classes are disjoint, hence no common instance can exist—i.e., no bank can be a lending institution according to this integration system.

Of course, there should be something wrong the way the two ontologies have been integrated, and this calls for a revision of the mappings. The engineer should either omit the mapping stating that mortgage lenders are necessarily profit companies, or the mapping stating that mortgage lenders are necessarily lending institutions. In both cases, the outcome will be acceptable by the engineer, and she/he should choose the one that best fits further analysis of the domain that she/he may have done after this unexpected discovery.

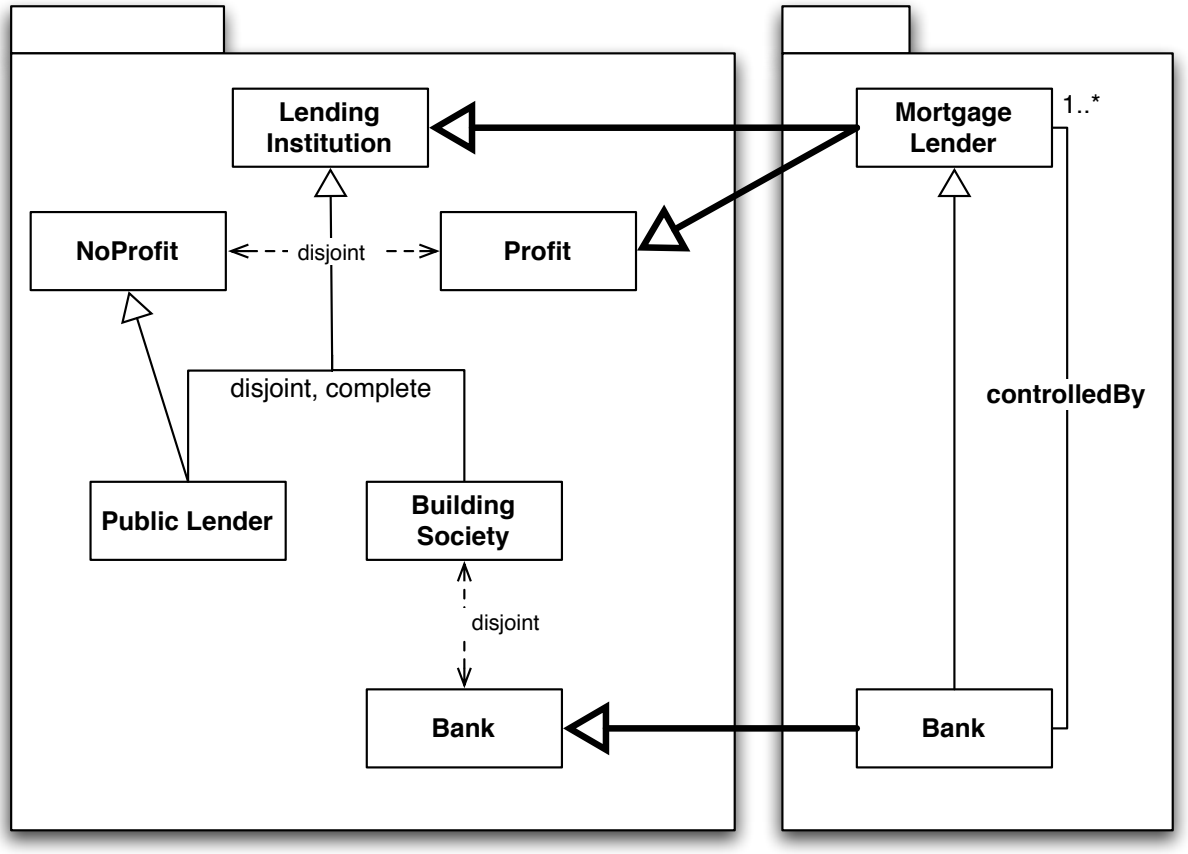

Figure 14: Ontology integration scenario.

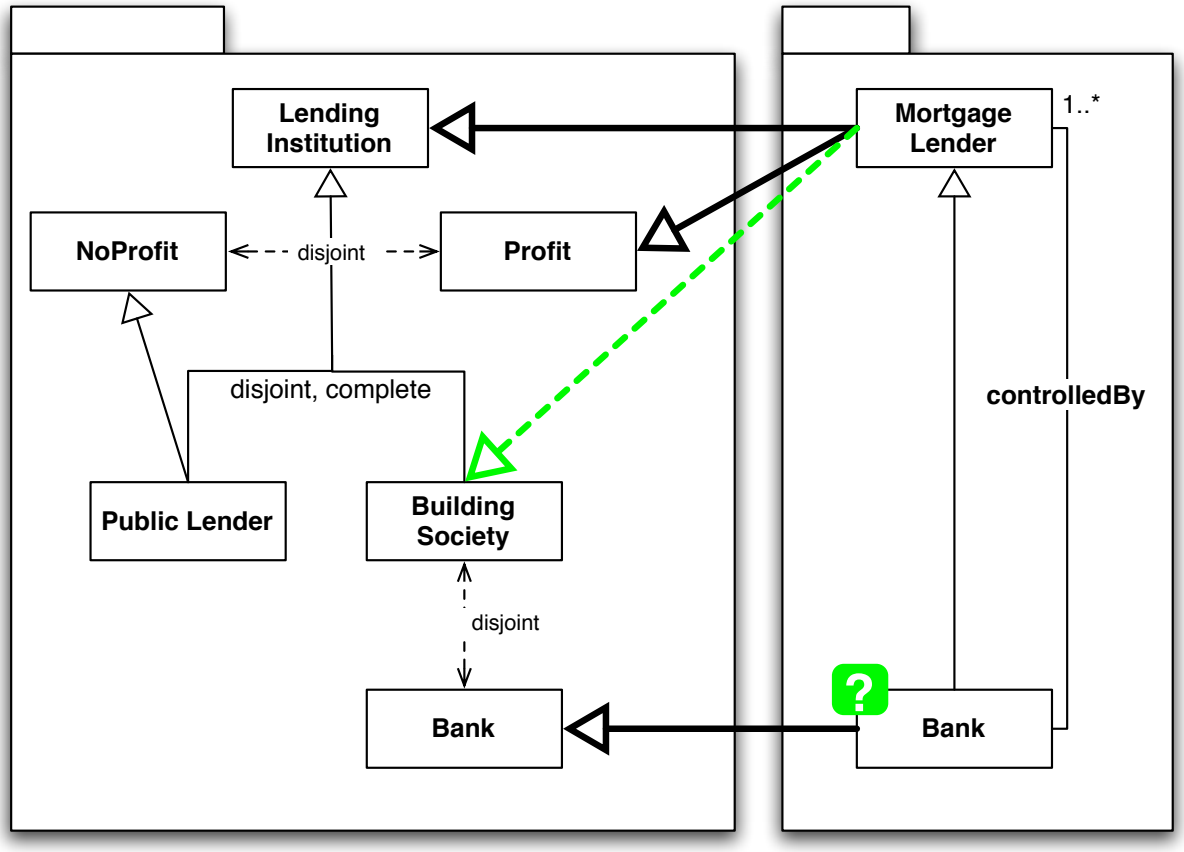

Figure 15: Ontology integration scenario with deductions after the integration process.

4.4 Ontology-mediated Interoperation

In this example we consider the problem of interoperability among different information sources. Let us consider three independent distributed databases depicted in Figure 16.

The first one (upper right in the figure) contains a database of mortgage lenders. The database contains information regarding the independence of a specific lender by means of a Boolean flag.

The second database (upper left) contains information about financial institutions, which are partitioned into banks and building societies (two disjoint classes).

The third database (bottom one) contains information about loan institutions, together with a binary relationship representing the fact that two institutions are associated. Note that this latter information is somewhat related to the "independence" flag of the first database.

The three databases are connected by means of the following three rules (depicted as double stroked arrows in the figure).

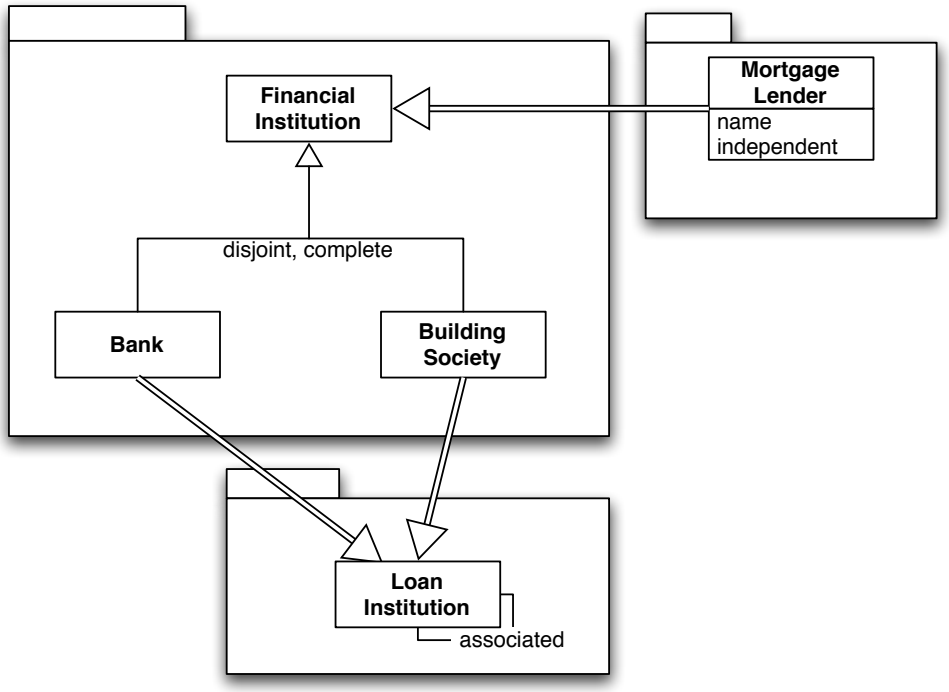

Figure 16: Ontology interoperability scenario.

$$kb_2 : \texttt{FinancialInstitution}(x) \Leftarrow kb_1 : \texttt{MortgageLender}(x, y)$$
$$kb_3 : \texttt{LoanInstitution}(x) \Leftarrow kb_2 : \texttt{Bank}(x)$$
$$kb_3 : \texttt{LoanInstitution}(x) \Leftarrow kb_2 : \texttt{BuildingSociety}(x)$$

We can assign at least two different semantics to the rules. The first one, is the so called Classical semantics, where the arrow is interpreted as logical implication. This is the semantics we implicitly adopted in the previous examples. Accordingly to this semantics the double stroked arrows can be seen as the thick arrows used in the previous diagrams; so it can be concluded that every mortgage lender in the first database should be a loan institution in the third database as well.

So, given the data

MortgageLender	
RBS	FALSE

FinancialInstitution

LoanInstitution
Leeds BS

the classical semantics would entail the virtual tables:

MortgageLender	
RBS	FALSE

FinancialInstitution
RBS

LoanInstitution
Leeds BS
RBS

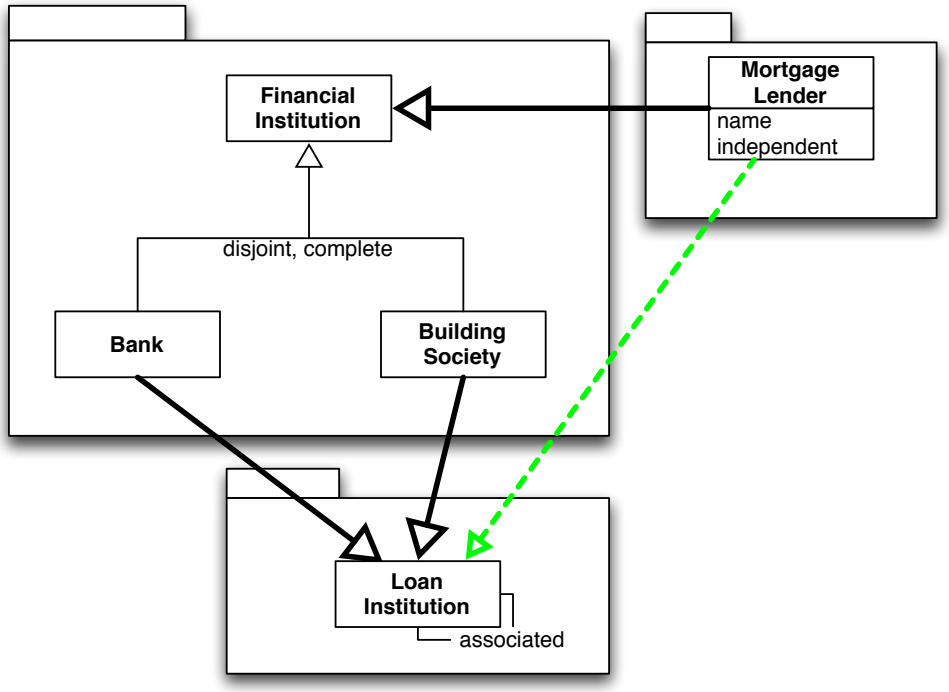

Figure 17: Ontology interoperability scenario with deductions under classical semantics.

This means that a reasoning enabled system can automatically derive the additional constraint shown in Figure 17.

However, we can interpret the rules in a Peer2Peer (P2P) fashion, where the arrows represent data flows between pair of database peers. We wont go into the semantic details in here; roughly speaking, rules represent queries (data fetching) over the corresponding peer. Moreover, the general idea is that they should be used locally *on-demand*; so, they should not affect the system as a whole. In fact, in our example, viewing the mappings under the Classical semantics introduces an implicit constraint between the first and the third database (see Figure 17) and, in a system of autonomous sources, this is not a desirable conclusion.

In this example, elements "migrating" in the second database into the class `Financial_Institution`, because of the first rule, are not known to be either banks or building societies. This means that the last two rules will transfer no data from the second to the third database; since no individual is known to be either definitely a bank (in which case the first rule would apply) or definitely a building society (in which case the second rule would apply).

So, given the data

MortgageLender	
RBS	FALSE

FinancialInstitution

LoanInstitution
Leeds BS

the P2P semantics would entail the virtual tables:

MortgageLender	
RBS	FALSE

FinancialInstitution
RBS

LoanInstitution
Leeds BS

consistently with the "data migration" metaphor: the RBS data migrated from `Mortgage Lender` to `Financial Institution`, but it can not migrate to `Loan Institution` due to the partitioning constraint.

Rules can be cyclic; to understand their expressive power suppose now (see Figure 18) to have an additional cyclic set of rules connecting the first and the third databases as follows:

$$kb_3 : \texttt{associated}(x, y) \Leftarrow kb_1 : \texttt{MortgageLender}(x, \mathbf{FALSE})$$
$$kb_1 : \texttt{MortgageLender}(x, \mathbf{FALSE}) \Leftarrow kb_3 : \texttt{associated}(x, y)$$
$$kb_1 : \texttt{MortgageLender}(y, \mathbf{FALSE}) \Leftarrow kb_3 : \texttt{associated}(x, y)$$

These rules serve the purpose of synchronising the data about loan institutions and mortgage lenders in the two databases. So, suppose we start with the following database:

MortgageLender	
RBS	FALSE

LoanInstitution
Leeds BS

In this example, the difference between the classical and the P2P semantics is evident only in presence of negative queries. In fact, in both cases the virtual database would be completed as follows:

MortgageLender	
RBS	FALSE
Leeds BS	*FALSE*

LoanInstitution
Leeds BS
RBS

Within the Classical semantics, a query to the third database asking for the non existence of some associated loan institution "Deutsche Bank" (i.e., a fresh new constant) will get a negative answer. Note that the semantics of a query is the certain answer semantics: in the classical case, there is at least one instance of the virtual database that contains "Deutsche Bank" as a load institution. However, adopting the P2P semantics, we actually expect a positive answer, since the only true information is the information that is fetched (or the local one). Please note that the intersection of all legal database instances in the classical case is still a legal database instance and it coincides with the unique legal instance in the P2P case (this is the instance obtained by migrating the data with a fixpoint).

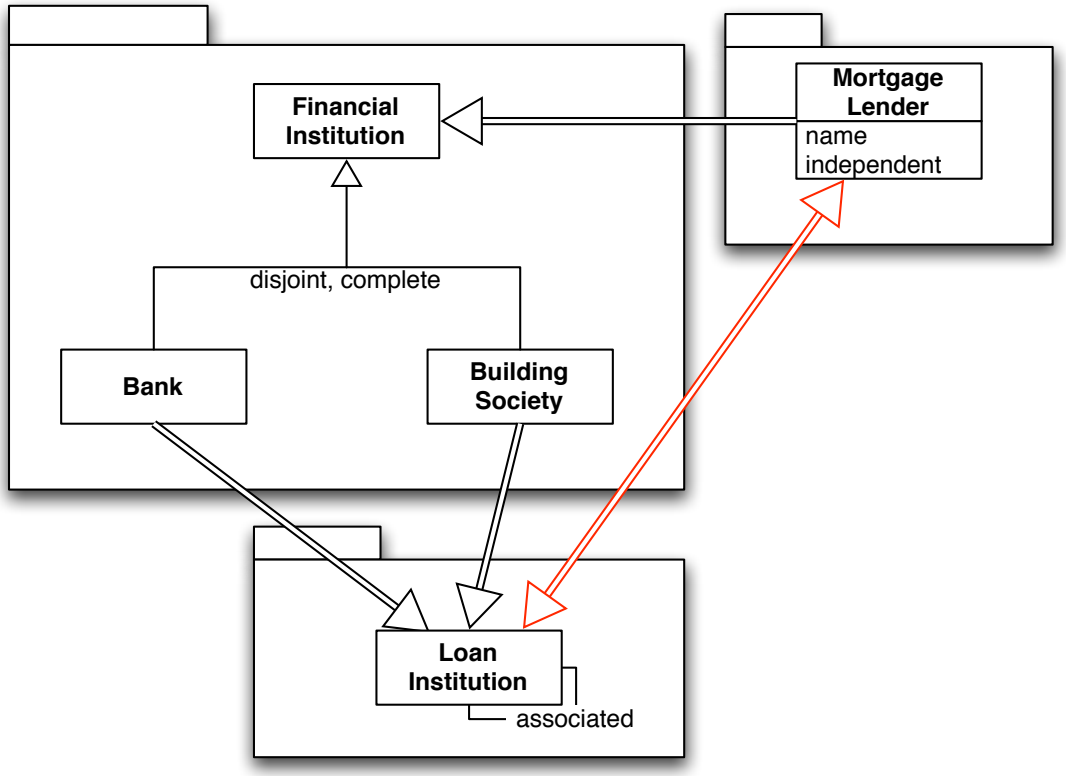

Figure 18: Ontology interoperability scenario with P2P semantics.

5 Conclusions

We have shown in this revised deliverable a categorisation of relevant use cases whose solution we suggest should involve ontologies and ontology-based tasks. This document serves as input for the discussion with the industrial partners of the TONES project, with the final goal of clearly identifying the relevant ontology-based tasks that will be studied in the TONES project.

References

[1] Russell Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

[2] José Júlio Alferes, Wolfgang May, and *al.* Use-cases on evolution. Deliverable I5-D2, Rewerse EU-IST Network of Excellence IST-2004-506779, February 2005.

[3] A. Calì, D. Calvanese, and *al.* State of the art survey. Deliverable D01, TONES EU-IST STREP FP6-7603, December 2005.

[4] David Chen and *al.* Practices, principles and patterns for interoperability. Deliverable D6.1, Interop EU-IST Network of Excellence IST-508011, May 2005.

[5] Tim Furche and *al.* Development of use cases, part i. Deliverable I4-D3, Rewerse EU-IST Network of Excellence IST-2004-506779, February 2005.

[6] S. Grimm, B. Motik, and C. Preist. Variance in ebusiness service discovery. In *Proceedings of the ISWC 2004 Workshop on Semantic Web Services*, 2004.

[7] L. Hotz and B. Neumann. Scene interpretation as a configuration task. *Künstliche Intelligenz*, 3:59–65, 3 2005.

[8] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. *Int. J. of Electronic Commerce*, 8(4):39–60, 2004.

[9] D.L. McGuinness and J.R. Wright. An industrial-strength description logic-based configurator platform. *IEEE Intelligent Systems*, 13(4):69–77, 1998.

[10] L. Nixon, M. Mochol, and *al.* Prototypical business use cases. Deliverable D1.1.2, KnowledgeWeb EU-IST Network of Excellence IST-2004-507482, January 2005.

[11] M.P. Shanahan. Perception as abduction: Turning sensor data into meaningful representation. *Cognitive Science*, 29:103–134, 2005.

[12] P. Shvaikol and *al.* Knowledge processing requirement analysis. Deliverable D1.1.3, KnowledgeWeb EU-IST Network of Excellence IST-2004-507482, January 2005.

[13] W. Siberski and *al.* Semantic web framework requirement analys. Deliverable D1.2.2, KnowledgeWeb EU-IST Network of Excellence IST-2004-507482, June 2005.

[14] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.