Ontology Reconciliation

Adil Hameed, Alun Preece, and Derek Sleeman

University of Aberdeen, Department of Computing Science,
Aberdeen AB24 3UE, UK
{ahameed|apreece|dsleeman}@csd.abdn.ac.uk

Summary. Ontologies are being applied very successfully in supporting information and knowledge exchange between people and organisations. However, for many reasons, different people and organisations will tend to use different ontologies. Therefore, in order to exchange information and knowledge, either everyone must adopt the same ontology — an unlikely scenario — or it must be possible to *reconcile* different ontologies. This chapter examines the issues and techniques in the reconciliation of ontologies. First, it examines the reasons why people and organisations will tend to use different ontologies, and why the pervasive adoption of common ontologies is unlikely. It then reviews alternative architectures for multiple-ontology systems on a large scale. A comparative analysis is provided of a number of frameworks which analyse types of mismatches between ontologies. The process of ontology reconciliation is outlined. Finally, some existing software tools that support reconciliation are surveyed, and areas are identified where further work is necessary.

1 Introduction

Ontologies are a key component of any information architecture, being an explicit specification of the conceptual model underpinning an information domain [5]. Regardless of the information architect's goal — whether it be, for example, the definition of an enterprise data model, the design of an organisation's Web site, or the creation of a corporate knowledge map — one of the first tasks is to elucidate and specify the conceptual and relational structures in the domain. For an enterprise data model, the ontology will identify aspects such as business processes, actors, and information objects [24]; for a Web site, the ontology specifies information categories, topics, labels, link types, and so on [21]; for a corporate knowledge map, the ontology defines the areas of expertise within the organisation, useful for documenting best-practices and experience, and also for "identifying who knows what" [20].

In providing a means of organising information and knowledge, ontologies also facilitate its communication and interchange. While the communication

and interchange of information and knowledge is not a necessary goal of creating an ontology — an individual may choose to create a "personal ontology" to organise their own information and knowledge [8] — it is generally seen as the most valuable use of ontology techniques [2]. However, as soon as ontologies are applied in supporting information and knowledge exchange between people and organisations, a fundamental problem arises: for many reasons, different people and organisations will tend to use different ontologies. Therefore, in order to exchange information and knowledge, either everyone must adopt the same ontology — an unlikely scenario — or it must be possible to *reconcile* different ontologies.

This chapter examines the issues and techniques in the reconciliation of ontologies. The next section examines the reasons why people and organisations will tend to use different ontologies, and why the pervasive adoption of common ontologies is unlikely. Section 3 then compares alternative architectures for multiple-ontology systems on a large scale. Section 4 reviews a number of frameworks which analyse types of mismatches between ontologies. The process of ontology reconciliation is outlined in Section 5. Finally, some existing software tools that support reconciliation (to at least some extent) are surveyed in Section 6. Section 7 concludes, identifying some areas where further work is necessary.

2 Living in a Multiple-Ontology World

The past decade has seen many long-term, large-scale efforts to develop standard, common ontologies to support information and knowledge sharing. These include work on domain-independent so-called *upper ontologies* such as Cyc [11] and SUO[1], and domain-specific ontologies such as the Enterprise Ontology [24] and the Engineering Mathematics ontology [5]. These efforts are undoubtedly important, and have led to a substantial level of maturity and agreement in the area of ontology technology and methodology.

Nevertheless, it is unrealistic to expect that in general all people and organisations developing information and knowledge application systems will use a common, shared ontology. There are several reasons for this. Firstly, at present there is often a competing choice of "common" ontology for a particular purpose. For example, Cyc and SUO offer alternative choices of upper ontology, while the Enterprise Ontology and TOVE [3] offer alternative business models. Even if, in the long run, particular ontologies become favoured, *de facto* or *de jure* standards, by that time there will likely be many "legacy" ontologies still in existence, which use an outmoded but still valid alternative. There will then be a need to reconcile such legacy ontologies with the standard common ontology.

A second reason for expecting continuing diversity in ontologies is that an ontology is often aligned with a particular *perspective* on the world. Whether

[1] `http://suo.ieee.org/`

it is a "personal ontology", designed to support an individual's needs and preferences, or an ontology created by a particular company to reflect that company's view on their industry, such ontologies will have biases and necessarily subjective features. If that individual or company needs to interchange information and knowledge with other individuals or organisations, there will be a need to reconcile multiple ontologies. In an environment of industrial knowledge management, it is easy to envisage a scenario where:

- individual workers need to reconcile their personal ontologies with the common ontology of their department, division, or company as a whole;
- a company needs to reconcile its ontology with those of other partner companies, or the industry as a whole.

Ontology reconciliation can therefore be at several levels: inter-personal, intra-organisational, and inter-organisational.

A third reason why the need for ontology reconciliation is unlikely to be eliminated is that, even if a single common standard ontology emerged that everyone committed to, this "überontology" would still need to *evolve* over time — conceptualisations of information and knowledge domains are not static. There will then be different versions of the "standard" in existence over time, and there will be a need to reconcile these different versions, to allow migration of information and knowledge between the versions.

These arguments are underscored by recent experience of working with information architecture for the World Wide Web. The Web is the largest and most significant information system in history; part of the reason for its success is that it is extremely tolerant of diversity in information modelling. Different Web sites use different information architectures; it is easy for individual people and organisations to "do their own thing", yet the Web as a whole is still usable and useful. Recent work in the context of developing a machine-processable, Semantic Web is acknowledging this diversity. The emerging layers of the W3C's architecture are incorporating support for a multiple-ontology Semantic Web, for example:[2]

- the lowest layers are founded on distributed information architecture standards: URIs and XML namespaces for creating "object identifiers" that can be defined with respect to a local ontology, yet referenced globally;
- the higher layers are tolerant of combining information from multiple ontologies (for example, RDF descriptions that refer to more than one RDF Schema) and articulating the relationships between ontologies (for example, OWL's `sameClassAs` property).

In the context of the Web, experience suggests that ontologies will proliferate and diverge — with the appearance of many personalised small-scale local conceptualisations — rather than converging on a few, large-scale standards under central control. The Semantic Web will work as a whole only if it is possible to reconcile its diversity of ontologies.

[2] `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`

3 Architecture

There exists a variety of alternative architectures for multiple-ontology systems, as shown in Figure 1. The simplest, "bottom up" approach, is merely to map between individual ontologies as needed. This architecture is shown in Figure 1(a). Here there is no attempt to identify common, standardised ontologies. Not all ontologies will be reconciled; only where there is a requirement to inter-relate particular individual ontologies. The advantages of this approach are its simplicity and flexibility: no overall common ontology is needed; there is great flexibility in being able to map only what's required, and mappings can be managed locally by the developers of specific individual ontologies. The most obvious disadvantage are that there will be many sets of mappings required when many ontologies need to be reconciled: $O(n^2)$ sets of (bidirectional) mappings for n individual ontologies in the worst case. Another significant disadvantage is that there is no general organising principle at work here: no attempt to identify common conceptualisations across the individual ontologies in a "top-down" fashion. For both of these reasons, this approach scales very poorly for large-scale ontology systems (of which the largest is the Semantic Web).

In contrast to the "bottom-up" approach, Figure 1(b) illustrates the approach where a single common, standard ontology is used as a basis for reconciling the individual ontologies. To map information and knowledge from one individual ontology O_1 to another individual ontology O_2, two steps are required: first, map from O_1 to the common ontology, then from the common ontology to O_2. Avoiding the direct mappings between individual ontologies cuts the number of sets of mappings down to just n (bidirectional) mappings for n ontologies, a significant improvement on the "bottom-up" approach. However, there will be a major cost in developing a common ontology with sufficient power to map all the individual ontologies; in some cases, this may not even be possible, so parts of some individual ontologies may still need to be mapped directly. Moreover, some flexibility is lost, in that mappings are no longer manageable locally, but only in relation to a centralised standard.

Figure 1(c) shows a variant of the common ontology approach, where there are a number of common ontologies, forming *clusters* of inter-related ontologies [26]. Each individual ontology maps to the common ontology for its cluster, and the common ontologies are mapped to allow the exchange of information and knowledge between the clusters. This is the most manageable, scalable approach in practice, as it combines the advantages of both the previous approaches: there is still a reduced number of mappings, and a principled approach to identifying common conceptualisations, as in Figure 1(b), yet also there is greater flexibility in managing mappings in a localised context, as in Figure 1(a). For these reasons, this third approach appears most likely to succeed in the context of the Semantic Web.

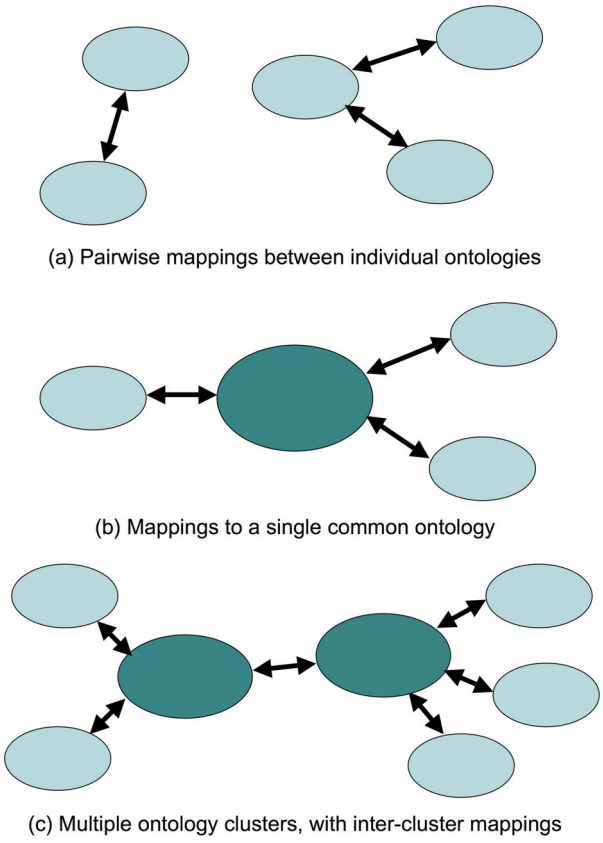

Fig. 1. Alternative ontology reconciliation architectures: ontologies are depicted as nodes, (bidirectional) mappings as arcs; dark-shaded nodes are common ontologies; light-shaded nodes are individual ontologies.

4 Ontology Mismatches

In order to reconcile ontologies, it is necessary to analyse the *mismatches* between individual ontologies. Mismatches might be present at a conceptual level, as well as at the terminological, taxonomical, definitional, and purely syntactic levels. It is necessary to detect and resolve such discrepancies, especially among the differing semantics. Correspondences may have to be established among the source ontologies, and overlapping concepts will need to be identified: concepts that are similar in meaning but have different names or structures, concepts that are unique to each of the sources [18]. This section surveys ontological mismatches from the perspective of researchers and

practitioners in three areas: knowledge representation [25], databases [27], and knowledge elicitation [22]. In each case, examples of real ontological mismatches are given, drawn from an empirical study of mismatches among ontologies of four different experts in the domain of Personal Computer (PC) advising [6]. In all the examples, terms in the different experts' ontologies are differentiated by prefixing them in a similar way to XML namespace prefixes; for example, the term `a:PC` identifies the concept `PC` in the ontology of Expert A, and `b:HardDisk` identifies the concept `HardDisk` in the ontology of Expert B.

4.1 Knowledge Representation Perspective

Visser et al [25] proposed a classification of ontology mismatches to explain semantic heterogeneity in systems. They distinguish *conceptualisation mismatches* and *explication mismatches* as the two main categories, described as follows:

Conceptualisation mismatches

These may arise between two or more conceptualisations of a domain. The conceptualisations could differ in the ontological concepts distinguished or in the way these concepts are related, as shown below:

- *Class mismatches* are concerned with classes and their subclasses distinguished in the conceptualisation:
  - A *categorisation mismatch* occurs when two conceptualisations distinguish the same class, but divide this class into different subclasses.
    Example:
    `c:Expert` *advises* `c:Staff` ∪ `c:Student`
    `b:Expert` *advises* `b:User` ∪ `b:MemberOfStaff` ∪ `b:Department` ∪ `b:Supplier` ∪ `b:Expert`
    These conceptualisations differ because the experts have partitioned the same class — as distinguished in their individual ontologies — into different aggregates of subclasses. The symbol ∪ signifies the union of the specified classes/concepts.
  - An *aggregation-level mismatch* occurs if both conceptualisations recognise the existence of a class, but define classes at different levels of abstraction.
    Example:
    `b:PC` → `b:Desktop` ∪ `b:Laptop`
    `c:PC` → `c:Desktop` ∪ `c:Tower` ∪ `c:Portable` ∪ `c:Server`
    The experts have identified the same (or similar) classes but defined them at dissimilar levels of abstraction. In particular, the second expert's notion of a "PC" is broader than that of the first expert — the second's includes servers, whereas the first's is restricted to users'

workstations. In this example, the symbol → denotes the relation 'is defined by'.

- *Relation mismatches* are associated with the relations distinguished in the conceptualisation. They concern, for instance, the hierarchical relations between two classes or, the assignment of attributes to classes:
  - A *structure mismatch* occurs when two conceptualisations perceive the same set of classes but differ in the way these classes are structured via relations.
    Example:
    `c:PC` *isMadeOf* `c:Part` ∪ `c:Component`
    `e:PC` *hasComponent* `e:Processor`
    Experts B and C have distinguished the same set of classes but differ in the way these classes are structured by means of the relations that associate their concepts. The following descriptions also reveal a difference in granularity of the domain semantics.
  - An *attribute-assignment mismatch* occurs when two conceptualisations differ in the way they assign an attribute to various classes.
    Example:
    `b:PC` *has* `b:Disk`, `b:PC` *has* `b:Space`
    `d:PC` *has* `d:Disk`, `d:Disk` *has* `d:Space`
    The ontologies differ in the way they assign an attribute to their respective subclasses. While expert B has assigned two disjoint attributes `Disk` and `Space` to the concept `PC`, expert D defined a hierarchical relationship between similar concepts.
  - An *attribute-type mismatch* occurs when two conceptualisations distinguish the same attribute, but differ in their assumed instantiations (range of possible value assignments).

Explication mismatches

These are not defined on the conceptualisation of the domain but on the way the conceptualisation is specified. They occur when two ontologies have different definitions, yet some component of the definition is identical. Visser et al distinguish three components of a definition: the *term* (T) used to denote a concept, the *definiens* (D) that comprises the body of the definition, and the underlying *concept* (C) being defined. Six different types have been specified, listed below. In the following examples, the → symbol denotes that the definiens which follow the arrow 'define' the term on the left-hand side of the description, and the ∧ operator is used to concatenate multiple definiens.

- *Concept (C) mismatch*: the definitions have the same terms and definiens, but differ conceptually. So, while the represented definitions are apparently identical, in fact they refer to different concepts.
  Example:
  `b:MinSpec` ← `b:Requirement` ∧ `b:PC` ∧ `b:User`

(referring to a user's hardware requirement)
`d:MinSpec` ← `d:Requirement` ∧ `d:PC` ∧ `d:User`
(referring to a system specification of software needed by user)
In this case, there are identical terms and definiens but each expert is referring to a quite different concept, revealed by the context of the definition as explained in parentheses below.

- *Concept & Definiens (CD) mismatch*: the definitions share the same term, but have different concepts and definiens. Apparently, different "things" are being defined, though the terms coincide.
  Example:
  `a:Spec` ← `a:Supplier` ∧ `a:StandardSpecsList`
  `b:Spec` ← `b:Specifies` ∧ `b:StandardSpecsList`
  `b:Spec` ← `b:Specifies` ∧ `b:Machine`
  `b:Spec` ← `b:Description` ∧ `b:Machine` ∧ `b:User`
  `c:Spec` ← `c:Requirement` ∧ `c:User`
  `c:Spec` ← `c:Requirement` ∧ `c:Application`
  `c:Spec` ← `c:Specification` ∧ `c:HardwareDevice`
  In these examples, an incidence of multiple descriptions for the same term implies that the expert gave distinct definitions in different contexts.

- *Definiens (D) mismatch*: the definitions have the same concept and the same term, but different definiens.
  Example:
  `b:FastPC` ← `b:ProcessorSpeed` ∧ `b:MemoryAmount`
  `c:FastPC` ← `c:CPUPentium4` ∧ `c:RAM64MegaBytes`

- *Term (T) mismatch*: the definitions share the same concept and the same definiens, but use different terms.

- *Concept & Term (CT) mismatch*: the definitions have the same definiens, but different concepts and terms. While the actual body of the definition is the same in each case, suggesting the same "thing" is being defined, the underlying concept and the term used for that concept is different.
  Example:
  `b:AdviceToUser` → `b:Expert` *advises* `b:User` *about* `b:Spec`
  `b:AdviceToSupplier` → `b:Expert` *advises* `b:Supplier` *about* `b:Spec`
  In Expert B's ontology, the concept 'advice' refers to the provision of a 'spec' (specification) to both 'users' and 'suppliers', albeit in different contexts.
  This is an instance of a discrepancy *within* Expert B's ontology.

- *Term & Definiens (TD) mismatch*: the definitions have the same concept, but dissimilar terms and definiens. This is essentially the exact opposite to the *C* mismatch — the same concept is represented completely differently in the two definitions.
  Example:

`b:MinSpec` ← `b:ProcessorP3` ∧ `b:Memory128MB`
`c:BasicPC` ← `c:CPUPentium` ∧ `c:RAM64MegaBytes`
Although they use different terms and definiens, both experts are referring to the same concept: a specification for an entry-level PC. This interpretation might be construed as subjective, but more domain-specific knowledge would be required to explicate the subtleties in such concepts.

4.2 Database Perspective

Wiederhold [27] contends that "data obtained from remote and autonomous sources will often not match in terms of naming, scope, granularity of abstractions, temporal bases, and domain definitions." Wiederhold's perspective is especially applicable in the context of mapping between ontologies and databases. His approach proposes the following types of data resource mismatches:

- *Key difference*: different naming for the same concept, for example synonyms.
  Example:
  `b:RMMidRangeSystemAcceleratorSpec2` (reference for customer)
  `b:GCAT03234` (reference for supplier & experts)
- *Scope difference*: distinct domains, or distinct coverage of domain members.
  Example:
  `b:Advice` (advice given by expert B to users, etc)
  `a:Advice` (technical advice sought by expert A from expert B)
  `a:Memory`, `d:Memory` (referring to RAM)
  `b:Memory`, `c:Memory` (referring to RAM and VRAM)
- *Abstraction grain*: varied granularity of detail among the definitions.
  Example:
  `c:FasterMachine` (referring to speed of computer)
  `b:FastestMachine` (referring to speed of CPU)
- *Temporal basis*: mismatches concerning 'time', for example monthly versus yearly income.
- *Domain semantics* (distinct domains, and the way they are modelled)
  Example:
  `b:Budget` (funds/financial outlay available to user)
  `b:Cost` (price of machine quoted by the supplier)
- *Value semantics*: differences in the encoding of values.

Wiederhold states that in order to 'compose' large-scale software there has to be agreement about the terms, since the underlying models depend on the symbolic linkages among the components [28].

4.3 Knowledge Elicitation Perspective

Shaw and Gaines [22] identified four distinct dimensions to map knowledge elicitation problems that are likely to occur when several experts are involved during the evolution of a knowledge-based system. Because experts 'work' with knowledge entities that comprise concepts and terms, ambiguities can arise among the way concepts are agreed upon. From this perspective, there are four possible cases:

- *Conflict*: the experts use the same term for different concepts.
  Example:
  `c:MinimumSpecification`: requirements of a certain hardware device
  `d:MinimumSpecification`: minimum system requirements for satisfactorily running a software package
  `a:Specifications`: referring to suppliers' specification lists
  `c:Specifications`: referring to (i) requirements of certain applications; (ii) user requirements; (iii) certain hardware devices (video cards and monitors)

- *Correspondence*: the experts use different terms for the same concept.
  Example:
  `b:Memory` versus `c:RAM`
  `b:Processor` versus `c:CPU`

- *Contrast*: the experts use different terms, and have different concepts.
  Example:
  `c:Staff` versus `b:ApprovedSuppliers`

- *Consensus*: the experts use the same term for the same concept. **Example:** `a:Monitor`, `b:Monitor`, `c:Monitor` — all refer to video display unit/screen

In each case with the exception of the last, *consensus*, there is a discrepancy. Shaw and Gaines developed a methodology and tools based on the repertory grid technique for eliciting, recognising and resolving such differences [22].

4.4 Comparing the Three Perspectives

Table 1 draws comparisons between the three perspectives in subsections 4.1, 4.2, and 4.3. It shows that the knowledge representation perspective (subsection 4.1) is essentially a finer-grained breakdown of the knowledge elicitation perspective (subsection 4.3), while the database perspective (subsection 4.2) is in some sense orthogonal to both of the others.

Knowledge elicitation	Database	Knowledge representation
Consensus (same name, same concept)	No mismatch — not considered	No mismatch — not considered
Conflict (same name, different concepts)	Examples of *scope difference*, *abstract grain*, and *temporal basis* given in subsection 4.2	Three cases: • C mismatch • CD mismatch • D mismatch
Correspondance (different names, same concept)	Examples of *key difference* given in subsection 4.2	Three cases: • T mismatch • CT mismatch • TD mismatch
Contrast (different names, different concepts)	Examples of *domain semantics* given in subsection 4.2	Not considered a mismatch

Table 1. Comparisons between mismatches identified in the knowledge elicitation perspective (subsection 4.3), the knowledge representation perspective (subsection 4.1), and the database perspective (subsection 4.2). In column 1, we are using the word "name" to refer to what is called "term" in column 3.

5 The Process of Ontology Reconciliation

Ontology reconciliation is generally a human-mediated process, although software tools can help (see Section 6). This is because most of the decisions on how to resolve the kinds of ontological mismatches surveyed in the previous section require a human to identify that different symbols represent the same concept, or that the same symbols represent different concepts. It is also a human's decision as to how to manage the reconciliation. There are three possibilities: *merging*, *aligning*, or *integrating*.

- *Merging* is the act of building a new ontology by unifying several ontologies into a single one [19, 23]. The ultimate goal is to create a single coherent ontology that includes all information from all the sources [18]. The new ontology is created from two or more existing ontologies with overlapping parts, and can be either virtual or physical [9].
- *Aligning* is used when sources must be made consistent and coherent with one another but kept separately [18]. It involves bringing two or more ontologies into mutual agreement, making them consistent and coherent [1, 9]. A set of alignment statements are created during this process, which collectively define the relationships between the original ontologies.
- *Integrating* entails building a new ontology by composing parts of other available ontologies [19]. Like merging, this process results in a new ontology. The difference between this approach and merging is that only parts

of the original ontologies will be integrated — the goal is not to achieve a complete merger.

Once a decison has been made on how to manage the reconciliation, the next step is to identify mismatches between the candidate ontologies. As an example, Figure 2 shows ontology fragments from one of the examples in Section 4.1, together with a fragment of a common ontology. Figure 2(a) and Figure 2(b) are drawn from human experts in the PC advising domain, while Figure 2(c) is based on a PC technology reference textbook. In each case, a fragment of the class hierarchy of types of computer is shown; super-classes are decomposed into their sub-classes, where the super-class is defined as the union of its respective subclasses.

Figure 3 and Figure 4 give the OWL definitions corresponding to the first two ontologies in Figure 2 (for an introduction to OWL, see Chapter **??**). Figure 3 is Expert B's ontology fragment, from Figure 2(a), and Figure 4 is Expert C's ontology fragment, from Figure 2(b).

There are several possible reconciliations between these ontology fragments. Figure 5 shows one set of alignments, where classes in the two experts' ontologies are equated with classes in the common ontology. In this reconciliation, the second expert's notion of a "PC" is considered broader than that of the first expert — the second expert's notion includes servers, whereas the first expert's notion is restricted to users' workstations. If a decision had been made to align the ontologies, then these class-equivalence relationships would constitute the set of reconciliation statements.

Figure 6 shows some additional reconciliation statements expressed in OWL, this time directly between the two individual ontology fragments from Expert B and Expert C (the statements for the reconciliations between the experts' ontologies and the common ontology shown in Figure 5 would be similar). Note how OWL's `sameClassAs` property is used to express the class-equivalence.

It is worth noting that, in a process of merging the two experts' ontologies, the common ontology, in fact, can be seen as one possible result of such a merger process, as it contains all of the information from the individual experts' ontologies.

6 Ontology Reconciliation Tools

A sizeable number of software tools has been developed to assist in the process of reconciling ontologies, although few of these have moved beyond the status of research prototypes. This section provides a short survey of a number of the better-known tools, as an illustrative snapshot of this evolving activity. Other notable tools include the Methontology project's WebODE [4], Ontolog-

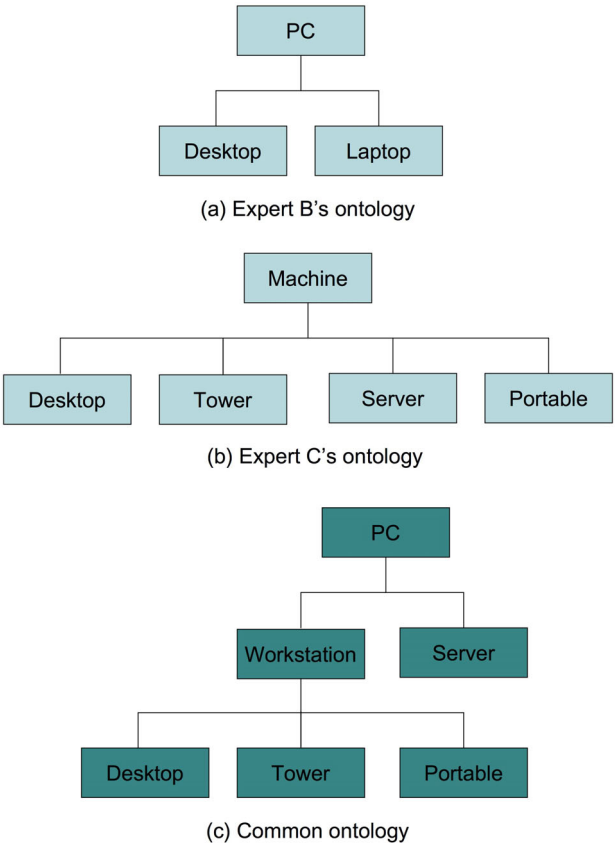

Fig. 2. Fragments of three alternative PC ontologies: those of individual experts B and C, and a common ontology for the PC domain. Each example shows super-classes decomposed into sub-classes, where the super-class is always the union of its subclasses.

ging/KAON [12], and ConcepTool [14]. A survey of ontology mapping tools has been undertaken by the European OntoWeb network[3]

6.1 Chimæra

Chimæra was developed by McGuinness et al at the Knowledge Systems Laboratory, Stanford University [13]. It is an interactive Web-based environment

[3] See deliverables of the "Enterprise Standard Ontology Environments" SIG at: http://www.ontoweb.org

```
<!DOCTYPE rdf:RDF [
  <!ENTITY reconto "http://www.csd.abdn.ac.uk/research/reconto">
]>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="&reconto;/samples/pc/expertb">

  <owl:Class rdf:about="PC">
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="Desktop"/>
      <owl:Class rdf:about="Laptop"/>
    </owl:unionOf>
  </owl:Class>
</rdf:RDF>
```

Fig. 3. OWL version of Expert B's ontology fragment, from Figure 2.

```
<!DOCTYPE rdf:RDF [
  <!ENTITY reconto "http://www.csd.abdn.ac.uk/research/reconto">
]>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="&reconto;/samples/pc/expertc">

  <owl:Class rdf:about="Machine">
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="Desktop"/>
      <owl:Class rdf:about="Tower"/>
      <owl:Class rdf:about="Server"/>
      <owl:Class rdf:about="Portable"/>
    </owl:unionOf>
  </owl:Class>
</rdf:RDF>
```

Fig. 4. OWL version of Expert C's ontology fragment, from Figure 2.

for merging and testing ontologies, which allows the user to bring together ontologies developed in different formalisms. The tool aims to support users in creating and maintaining distributed ontologies on the web. Chimæra uses the Ontolingua ontology editor, and is OKBC-compliant. Its Web-based user interface is simply HTML, augmented with JavaScript.

The two major functions supported by Chimæra are merging multiple ontologies, and evaluating ontologies with respect to their coverage and cor-

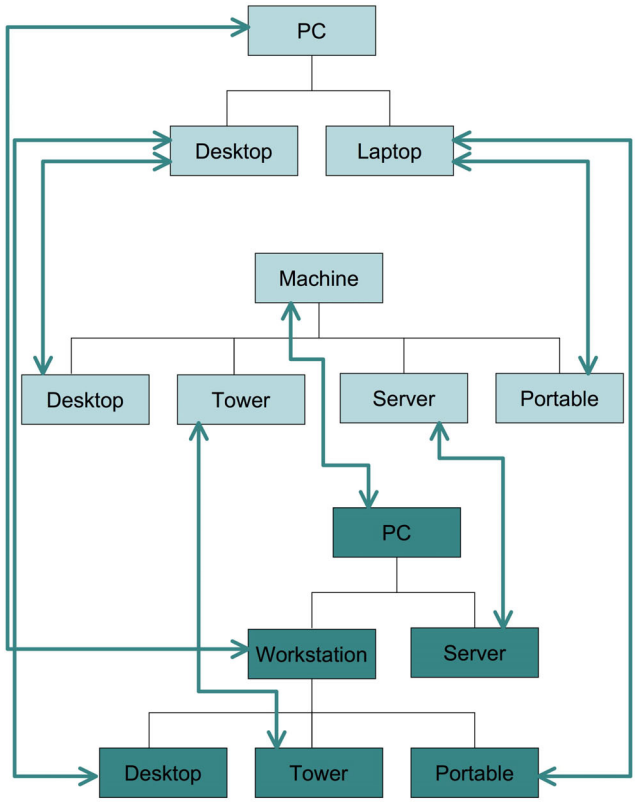

Fig. 5. Possible alignments between the three PC ontology fragments from Figure 2, showing classes in the two experts' ontologies equated with classes in the common ontology; the common ontology can be viewed as a merger of the two individual experts' ontologies.

rectness. Chimæra considers the task of merging to be one of combining two or more ontologies that may use different vocabularies and may have overlapping content. The task of evaluating single or multiple ontologies is addressed by producing a test suite that evaluates (partial) correctness and completeness of the ontologies. This involves finding and reporting provable inconsistencies, possible inconsistencies, and areas of incomplete coverage. The tool has other features like loading knowledge bases in differing formats, reorganising taxonomies, resolving name conflicts, browsing ontologies, editing terms, and so on.

```
<!DOCTYPE rdf:RDF [
  <!ENTITY reconto "http://www.csd.abdn.ac.uk/research/reconto">
  <!ENTITY expertb "/samples/pc/expertb#">
  <!ENTITY expertc "/samples/pc/expertc#">
]>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="&reconto;/samples/pc/articulation">

  <rdf:Description rdf:about="&reconto;&expertb;Desktop">
    <owl:sameClassAs rdf:resource="&reconto;&expertc;Desktop"/>
  </rdf:Description>

  <rdf:Description rdf:about="&reconto;&expertb;Laptop">
    <owl:sameClassAs rdf:resource="&reconto;&expertc;Portable"/>
  </rdf:Description>
</rdf:RDF>
```

Fig. 6. OWL version of some alignments directly between Expert B's ontology fragment, and Expert C's ontology fragment.

Users can request analysis or guidance during the merging process. The tool will then point to the places in the ontology where attention is required. In its suggestions, Chimæra mostly relies on which ontology the concepts came from and, for classes, on their names. For example, it will point a user to a class in the merged ontology that has two slots derived from different source ontologies, or that has two sub-classes that originated in different ontologies. Help offered during the merging process includes:

- Generation of a 'name resolution' list that helps the user in the merging task by suggesting terms each of which is from a different ontology that are candidates to be merged.
- Generation of a 'taxonomy resolution' list where the tool suggests taxonomy areas that are candidates for reorganization. A number of heuristic strategies are used for finding such points for taxonomies. Currently, the tool implements only partial support for the merging of class-subclass taxonomies.

In evaluating ontologies with respect to their coverage and correctness, Chimæra provides simple checks for incompleteness, syntactic analysis, simple taxonomic analysis, and some semantic evaluation.

Some limitations of Chimæra in its current form are that it leaves ontology reconciliation decisions entirely to the user, and does not make any suggestions itself. Also, use of Chimæra is hampered by very slow performance, a non-intuitive user interface, and a steep learning curve. Some of these weaknesses

stem from the tool's Web-based architecture, which limits performance and user interface sophistication.

Ongoing work on Chimæra includes extending reasoning capabilities, providing semantic analysis in the reconciliation process, offering greater extensibility, and opening up the tool's usability to non-experts.

6.2 ONION

The ONION (ONtology CompositION) system was developed by Mitra, Wiederhold, and colleagues in the Stanford University Database Group [16]. The ONION project proposes a scalable and maintainable approach based on interoperation of ontologies. The motivation for the work is to handle distributed queries crossing the boundaries of underlying information systems. For this to be possible, the interoperation between the ontologies for the individual information systems need to be precisely defined.

The ONION approach includes an algebra for knowledge composition, featuring operators such as union, intersection, and difference. Articulation of ontology interdependencies are expressed using this algebra. The articulations essentially form a graph of ontology inter-relations, similar to the reconciliations illustrated earlier in Figure 5 (in the figure, the only articulation shown is class-equivalence).

To assist in creating the articulations, the ONION tool takes two sets of concepts and matches them using dictionaries (in particular, WordNet [15]) and information retrieval techniques (specifically, use of a corpus of documents). While the tool suggests possible articulations, it is left to the expert to define their chosen articulations. Input to the articulation-generation tool is in RDF, and the tool's output is plain text.

6.3 OntoView

OntoView is a tool under development by Klein et al at the Vrije University, Amsterdam, and OntoText, Sofia [10]. OntoView offers Web-based ontology versioning. In its current form, the OntoView tool is a slightly enhanced version of CVSWeb (Concurrent Versions System), an open-source network-transparent version control system. At its core, the tool identifies changes among different ontology versions. The comparison function is inspired by the UNIX *diff* (difference) file-comparison utility. However, OntoView compares ontologies at a structural level instead of line-level, showing which definitions of ontological concepts or properties have changed.

The tool is currently being re-implemented on top of a custom-designed ontology repository system. The OntoView project promises to provide a transparent interface to arbitrary versions of ontologies. It is planned that an internal specification of the relations between the different variants of ontologies will be built. This specification is based on the versions of ontologies themselves, on explicit change specifications, and on additional human input.

Users will be able to differentiate between ontologies at a conceptual level and to export the differences as transformations or adaptations.

OntoView is still being developed; however, its current and proposed features include:

- read in ontologies, ontology updates, adaptations and/or mappings
- view a specific version or variant of an ontology
- provide a unique and persistent identification of versions
- allow users to assign properties to differences (type, etc)
- automatically perform inconsistency checks of version combinations
- differentiate ontologies:
  - show changed formal definitions
  - show changed comments
  - show type of change: conceptualisation or explication
- export translation/transformations/adaptations.

6.4 Prompt

The Prompt tool was developed by Noy and Musen within the Medical Informatics group at Stanford University [17]. Prompt offers a semi-automatic, interactive ontology-merging tool, which guides a user through the merging process, making suggestions, determining conflicts, and proposing resolution strategies. Prompt is implemented as a plug-in to the Protégé-2000 integrated knowledge-base editing environment. Protégé-2000 provides an extensible architecture for the creation of customised knowledge-based tools, and is compliant with the OKBC standard for frame-based KBs.[4]

The Prompt merging process aims to find 'semantically-similar' terms, although it does this through syntactic analysis. Initial suggestions are based on the lexical similarity of the frame names. The tool identifies candidates for merging as pairs of 'matching terms' — terms from different source ontologies representing similar concepts. Prompt suggests merging identical classes in two ontologies, where these classes are denoted by lexically-identical tokens. A graphical user interface helps users carry out interactive merging.

Prompt is intended to determine not only syntactic but also semantic matches based on (i) the content and structure of the source ontologies (for example, names of classes and slots, sub-classes, super-classes, domains and ranges of slot values) and (ii) the user's actions (that is, incorporating in its analysis the knowledge about the similarities and differences that the user has already identified). This algorithm relies on limited input from the user. The user does not need to analyse the structure of the ontology deeply, just to determine some pairs of terms that "look similar". As such, Prompt will only be useful in cases where the candidate ontologies already have close similarities.

[4] `http://protege.stanford.edu`

An enhanced version of Prompt, Anchor-Prompt, promises a number of extensions, including consideration of similarity between class hierarchies, and the use of similarity scores in determining the potential match between classes.

7 Conclusion and Way Forward

As ontology usage becomes more prevalent, the need for ontology reconciliation increases. This chapter has demonstrated that there is now a mature understanding of the kinds of mismatches that occur between different ontologies. Examining a representative sample of the software tools currently available to assist in the reconciliation process shows that, while a number of promising tools are already available, these are largely in the form of laboratory prototypes, and that more work is needed in this area. Moreover, it is unlikely that one single tool will ever emerge that satisfactorily handles all aspects of ontology reconciliation. For this reason, Hameed et al propose in [7] the creation of a *workbench* within which a variety of tools can be harnessed to assist ontology engineers in performing reconciliations. Figure 7 illustrates this proposed system.

The proposed workbench will guide a user in selecting appropriate tools for the kinds of mismatch identified, where a tool is selected on the basis of the kinds of mismatch it tackles, and also the knowledge representation formalisms on which it operates. Integrating the available — and newly emerging — tools in such a coherent framework, in the context of a systematic approach to identifying ontological mismatches, will be a significant step in managing the ontology reconciliation problem.

References

1. Corcho, O, Gómez-Pérez, A (2001) Solving Integration Problems of E-Commerce Standards and Initiatives through Ontological Mappings. *IJCAI-01 Workshop on Ontologies and Information Sharing*, pages 131–140
2. Fensel, D (2000) *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag
3. Fox, M, Gruninger, M (1998) Enterprise Modelling. *AI Magazine*, Fall, 109–121
4. Gómez-Pérez, A, Moreno, A, Pazos, J, Sierra-Alonso, A (2000) Knowledge Maps: An essential technique for conceptualisation. *Data and Knowledge Engineering*, **33**(2), 169–190
5. Gruber, T R (1993) A Translational Approach to Portable Ontology Specifications. *Knowledge Acquisition*, **5**, 199–220
6. Hameed, A, Sleeman, D H, Preece, A (2001) Detecting Mismatches in Experts' Ontologies through Knowledge Elicitation. In Bramer, M, Coenen, F, Preece, A (eds), *Research and Development in Intelligent Systems XVIII*, Springer-Verlag, pages 9–22

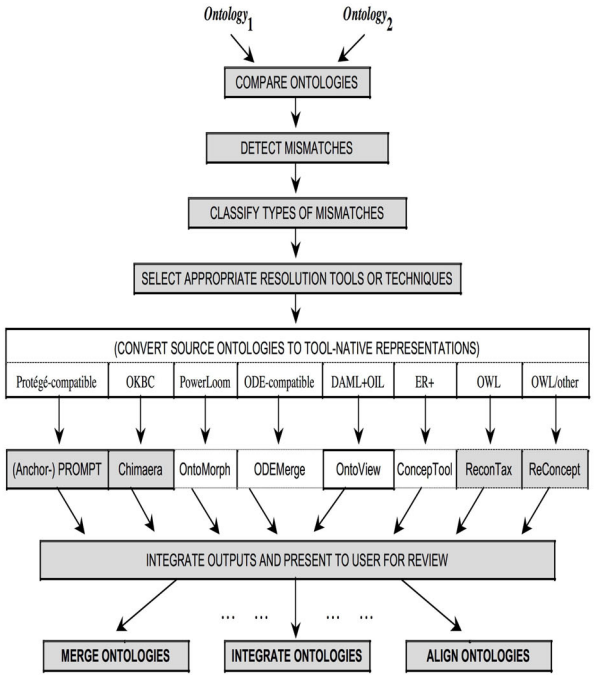

Fig. 7. Proposed workbench architecture for managing ontology reconciliations, by harnessing existing — and new — tools. The shaded components are within the scope of the current implementation; other components are planned for future expansion.

7. Hameed, A, Sleeman, D H, Preece, A (2002) OntoManager: A Workbench Environment to facilitate Ontology Management and Interoperability. In *EON-2002: EKAW-2002 Workshop on Evaluation of Ontology-based Tools*
8. Huhns, M N, Stephens, L M (1999) Personal Ontologies. *IEEE Internet Computing*, **3**(2) 85–87
9. Klein, M (2001) Combining and relating ontologies: an analysis of problems and solutions. *IJCAI-01 Workshop on Ontologies and Information Sharing*, pages 53–62
10. Klein, M, Kiryakov, W, Ognyanov, D, Fensel, D (2002) Ontology Versioning and Change Detection on the Web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, Sigüenza, Spain

11. Lenat, D, Guha, R (1990) *Building Large Knowledge-Based Systems*, Addison Wesley, Reading
12. A. Maedche, A, Motik, B, Stojanovic, L, Studer, R, Volz, R (2002) Managing Multiple Ontologies and Ontology Evolution in Ontologging. In *Proceedings Conference on Intelligent Information Processing (IIP2002)*, Kluwer
13. McGuinness, D L, Fikes, R, Rice, J, Wilder, S (2000) An Environment for Merging and Testing Large Ontologies. In Cohn, A, Giunchiglia, F, and Selman, B (eds), *KR2000: Principles of Knowledge Representation and Reasoning*, pages 483–493
14. Meisel H, Compatangelo, E (2002) EER-CONCEPTOOL: a "Reasonable" Environment for Schema and Ontology Sharing. In *Proc. of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2002)*, IEEE Computer Society Press, pages 527–534
15. Miller, G A (1995) WordNet: a Lexical Database for English. *Communications of the ACM*, **38**(11), 39–41
16. Mitra, P, Kersten, M. Wiederhold, G (2000) Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Proceedings of the 7th Int. Conf. on Extending Database Technology*, Springer-Verlag
17. Noy, N F, Musen, M A (2001) Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In *Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, USA
18. Noy, N F, Musen, M A (2000) PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *IJCAI-01 Workshop on Ontologies and Information Sharing*, pages 63–70
19. Pinto, H S, Martins, J P (2001) A Methodology for Ontology Integration. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001)*, ACM Press
20. Preece, A, Sleeman, D H, Flett, A N, Curry, D, Meaney, N, Perry, P (2001) Better Knowledge Management through Knowledge Engineering. *IEEE Intelligent Systems*, **14**(1), 26–36
21. Rosenfeld, L, Morville, P (2002) *Information Architecture for the World Wide Web*, OReilly
22. Shaw, M L G, Gaines, B R (1989) Comparing Conceptual Structures: Consensus, Conflict, Correspondence and Contrast. *Knowledge Acquisition*, **1**(4), pp. 341–363
23. Stumme, G, Maedche, A (2001) Ontology Merging for Federated Ontologies on the Semantic Web. *IJCAI-01 Workshop on Ontologies and Information Sharing*, pages 91–99
24. Uschold, M, King, M, Moralee, S, Zorgios, Y (1998) The Enterprise Ontology. *Knowledge Engineering Review*, **13**.
25. Visser, P R S, Jones, D M, Bench-Capon, T J M, Shave, M J R (1997) An Analysis of Ontology Mismatches; Heterogeneity vs. Interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA.
26. Visser, P R S, Tamma, V A M (1999) An Experiment with Ontology-Based Agent Clustering. In *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, Stockholm, Sweden.
27. Wiederhold, G (1992) Mediators in the Architecture of Future Information Systems. *IEEE Computer*, March.

28. Wiederhold, G (1994) An Algebra for Ontology Composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, September.