# Detecting Mismatches among Experts' Ontologies acquired through Knowledge Elicitation

Adil Hameed, Derek Sleeman, Alun Preece
Department of Computing Science
University of Aberdeen
Aberdeen, Scotland, UK
{ahameed, dsleeman, apreece}@csd.abdn.ac.uk
www.csd.abdn.ac.uk

**Abstract:** We have constructed a set of ontologies modelled on conceptual structures elicited from several domain experts. Protocols were collected from various experts who advise on the selection/ specification and purchase of PCs. These protocols were analysed from the perspective of both the processes and the domain knowledge to reflect each expert's inherent conceptualisation of the domain. We are particularly interested in analysing discrepancies within and among such *experts' ontologies*, and have identified a range of ontology mismatches. A systematic approach to the analysis has been developed; subsequently we shall develop software tools to support this process.

## 1. Introduction

An *ontology* is an explicit specification of a conceptualisation and is described as a set of definitions of content-specific knowledge representation primitives: classes, relations, functions, and object constants [6]. A collection of such conceptual models, when rich with domain-oriented content, can enable interoperability between systems, facilitate communication amongst people and organisations, and make existing knowledge shareable and reusable. Ontologies are also becoming increasingly important in internet applications as they can enhance the functioning of the World Wide Web in many ways: from providing 'meaning' for annotations in Web pages to empowering innovative services over the emerging Semantic Web [2]. Due to the distributed nature of ontology development, multiple ontologies covering overlapping domains is now the norm, and reconciliation is a vital issue.

Hitherto, most ontologies have been constructed as abstractions over existing software artefacts (viz., knowledge bases, databases, etc.) or built from published/documented reference sources. There is little evidence in the literature with regard to building and managing *experts'* ontologies – inherent conceptualisations elicited directly from human experts. We believe management of experts' ontologies is an important and as yet overlooked issue. The paper describes an approach we have evolved to construct such ontologies. We commence at the very beginning of the knowledge acquisition (KA) cycle and elicit

domain knowledge from several human experts. A systematic analysis leads to the formalisation of distinct ontologies that essentially are effective models of each expert's intrinsic conceptual structures. These abstractions then arguably are ideal for enabling the sharing and reuse of the experts' domain and task knowledge in a heterogeneous environment.

## 1.1 Ontology Mismatches

Utilisation of multiple ontologies with an objective to share/reuse knowledge, even within a common domain, could be hampered by the fact that they may not conform to one another. Inconsistencies might be present at a conceptual level, as well as at the terminological and definition level. It is necessary to detect and resolve such discrepancies, especially among the shared semantics. Correspondences may have to be established among the source ontologies, and overlapping concepts would 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 [10]. AI and database researchers have been working on converging ways to identify and resolve inconsistencies that occur in data/knowledge. There is general agreement that formal representations such as ontologies and schemas are necessary to tackle this problem. We carried out a comparative study of three distinct approaches:

**Visser *et al.*** [15] have 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 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;
  - An *Aggregation-level mismatch* occurs if both conceptualisations recognise the existence of a class, but define classes at different levels of abstraction.
- *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;
  - An *Attribute-assignment mismatch* occurs when two conceptualisations differ in the way they assign an attribute (class) to other classes;
  - An *Attribute-type mismatch* occurs when two conceptualisations distinguish the same (attribute) class but differ in their assumed instantiations.

Explication mismatches 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 where their terms (T), their definiens (D), or their ontological concepts (C) are identical. Six different types have been specified:

- *Concept & Term* (CT) *mismatch* (same definiens, but differ in concepts & terms)
- *Concept & Definiens* (CD) *mismatch* (same term, different concept & definiens)
- *Concept* (C) *mismatch* (same terms & definiens, but differ conceptually)
- *Term & Definiens* (TD) *mismatch* (same concept, dissimilar terms & definiens)
- *Term* (T) *mismatch* (same concept, same definiens, but different terms)
- *Definiens* (D) *mismatch* (same concept, same term, but different definiens)

**Wiederhold** [16] 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." He has therefore proposed the following types of data resource mismatches:

- *Key difference* (different naming for the same concept, e.g. synonyms)
- *Scope difference* (distinct domains: coverage of domain members)
- *Abstraction grain* (varied granularity of detail among the definitions)
- *Temporal basis* (concerning 'time', e.g., monthly budget versus family income)
- *Domain semantics* (distinct domains, and the way they are modelled)
- *Value semantics* (differences in the encoding of values)

He 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 [17].

**Shaw & Gaines** [12] have 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. For instance, experts may use:

- the same term for different concepts (*Conflict*),
- different terms for the same concept (*Correspondence*),
- different terms and have different concepts (*Contrast*).

Only when they use the same term for the same concept (Consensus) would there be no discrepancy. The authors have also developed a methodology and tools based on the Repertory Grid technique for eliciting, recognising and resolving such differences.

In order to detect factual instances of such mismatches, we have hand-crafted a set of domain ontologies that represent five distinct conceptualisations – the first four were modelled on experts, and the fifth was built from industry-standard sources of reference [4] and [13]. The area of personal computer (PC) configuration/ specification was chosen as an approachable domain. In the following section, we describe how knowledge was acquired from the several domain experts; the third section gives some examples of the kinds of ontological mismatches we have detected. These mismatches have been related to the three approaches cited above. In the last section, we conclude with an overview of our ongoing and further work.

## 2. Knowledge Acquisition

Knowledge elicitation was carried out in two phases: semi-structured interviews in phase 1 provided protocols, and structured tasks in the second phase enabled us to build glossaries of domain concepts. After each phase, analysis was done to refine the elicited artefact [5]. A preliminary analysis of the protocols also gave us an insight into the problem-solving strategies employed by each expert. Further analyses were carried out to confirm the outcomes prior to the formulation of ontologies. Conceptual graphs were used to model the relationships among domain concepts. A coding scheme was developed for this purpose, and the knowledge engineer was able to apply it consistently, after an independent coder verified the procedure.

### 2.1 Phase 1: Interviews

We interviewed experts whose technical proficiency is in personal computer systems and who regularly specify and configure PCs for a variety of users. The specification for a PC configuration usually consists of a list of standard components or parts, each of which can have certain specifiable attributes. Standardisation of configurations is an important objective not just for vendors or suppliers who are forced to compete in a demanding market; but also for large corporate buyers who want to keep the cost of new acquisitions and periodic upgrades down.

The questions posed were designed to elicit terms and concepts that the experts use on a regular basis to describe the various domain entities. These, along with the context in which they are used could be said to represent the expert's conceptual model of the domain. We got the experts to discuss several cases where they have advised users about purchasing a suitable PC. Typically, the expert would determine from the prospective user/customer the kind of applications they would be working on and/or ascertain the type of software they would want to run. S/he would then evolve a suitable hardware configuration to meet those requirements.

### 2.2 Preliminary Analysis

We have interviewed four domain experts in detail, over two sessions, and we were able to obtain an understanding of the inferencing processes 'applied' by the expert, and the distinctive approach taken by each of them for arriving at a suitable PC configuration. The initial analysis provided us with:

- A description of the process (or processes) that each expert follows in order to specify suitable PC configurations. These have been interpreted by the knowledge engineer and presented in the form of flow-charts and schematic diagrams to retain the accuracy of the expert's inferencing.
- The domain vocabularies used by the experts, which consist of terms and definitions that describe their inherent concepts. These have been organised as glossaries.

A technical report has been published [7] with a complete record of the knowledge elicitation sessions in the form of experts' protocols which were transcribed verbatim from audiotapes. Also included are the results of the analyses and a comprehensive set of glossaries extracted from the protocols.

### 2.2.1 Expert A

Expert A follows a pre-determined procedure to resolve queries from prospective customers. She helps customers select a suitable machine, from a current 'standard specifications' list of one of the approved suppliers. If additional upgrades or add-ons are required, then the buyer can select appropriate items from the set of standard options for the particular machine chosen. The expert ensures that the supplementary items are available with the supplier, viz., extra memory, peripheral devices, or add-ons such as modems, high-resolution monitors, Zip drives. She also makes certain that the total price is within the customer's budget. In case this expert cannot facilitate the selection of an acceptable configuration, or if there are any technical or application-related issues that cannot be resolved easily, she refers these 'non-regular' cases to Expert B, who is her line manager. The problem solving strategy used by Expert A can be described as a 'selection' algorithm [11].

### 2.2.2 Expert B

Most of the users (prospective customers) that approach Expert B for advice are the so-called 'non-regular' cases. Expert B's decision-making is based around a set of 'standard' system specs that he has devised. These are a starting point of reference. An evaluation of his protocols revealed that the expert follows a process-oriented approach, which consists of a series of distinct tasks/sub-tasks. According to the expert, his reasoning process is not strictly procedural. Instead, he adopts a holistic approach to determine the most suitable PC configuration that will meet the prospective customer's needs. After analysing several protocol segments, it became evident that the expert is following a 'hybrid' inferencing strategy. His inferences lead him on a goal-directed and forward-chaining path, but at times he backtracks when any of the user requirements, budgetary or technical constraints are not satisfied. The following steps model his decision-making process:

a). Begin with the user given requirements (natural language semi-technical phrases);

b). Identify 'key' terms, like processor, memory, graphics…;

c). If terms are imprecise or vague, resolve ambiguities by eliciting more information from the user until requirements are clearer;

d). Also, determine 'functional requirements', viz., understand what kind of tasks the user plans to perform with the PC, which applications would be run, etc.;

e). Ascertain user's budgetary constraints;

f). Select from the set of standard specs, a suitable system (a product/'model' from the supplier's list) that matches most closely the user's requirements and budget;

g). If additional enhancements or upgrades are necessary, then make suitable changes to the relevant components in the specs, keeping track of the budgetary constraint as well as any technical limitations;

h). Repeat step (g) until user and constraints are satisfied.

We believe that this expert's reasoning strategy essentially corresponds to the 'selection' and 'classification' PSM (problem-solving method) as described in [11].

### 2.2.3 Expert C

Unlike Experts A and B, this expert does not have a pre-determined set of 'standard' system 'specs'. He usually begins specifying a new configuration by starting with the current 'mid-range' system available, and working his way up (or down, as the case maybe), to arrive at a suitable specification. He constantly reassesses the mid-range system by keeping in touch with the state-of-the-art developments in the PC hardware market. The key constraint he bears in mind is the buyer's budget. Depending on how much money the buyer has to spare the expert either enhances or downgrades the 'mid' specification until all the user's requirements are met. The overall procedure employed is a form of the 'classification' algorithm [3] (selection & refinement) [11].

### 2.2.4 Expert D

Expert D believes that the 'right' way to configure a PC is to begin with a clear understanding of the actual application needs of the prospective user(s). He determines what 'tasks' the user wants to perform. After ascertaining this, he identifies suitable software applications (or packages) that will enable the user to perform these tasks. The expert believes it is necessary to agree on all requisite software before a hardware configuration can be specified. This is carried out in an iterative manner as follows:

a). The minimum system requirements for each software package are obtained from the software publisher;

b). The hardware requirements are then evaluated to determine which application is most greedy, i.e., most demanding, in terms of each specific component. Viz., CPU speed, minimum requirement for main memory, amount of disk space that is necessary, and any specific input/output pre-requisites;

c). Each component, along with its highest measure of requirement, is included on a specification list.

The final iteration would result in the configuration of a machine that would safely run all necessary software packages, thereby ensuring the user can perform all essential tasks. Some leeway may also be provided by enhancing the final configuration in order to 'future-proof' the machine against obsolescence of technology and also to accommodate newer or more advanced versions of software. An algorithm based on the 'classification' [3] and 'selection' methods [11] can best describe the inferencing strategy employed by Expert D with emphasis being placed on the acquisition of the *requirements*, which the machine must meet.

## 2.3 Phase 2: Structured Elicitation

Although interview is by no means the sole technique in determining conceptual structures, it was deemed an appropriate initial approach towards eliciting domain knowledge from the experts. As such, it is significant that in the first phase of knowledge elicitation, we were able to gain a clear understanding of the strategies/PSMs employed by each expert. Based on these initial results, we administered the following task-based experiments in phase 2 to unambiguously elicit the experts' domain terminology:

### 2.3.1 Step 1: Eliciting Terms, Definitions & Relations

A collage consisting of several graphic images of the PC (external & internal views) was shown to the experts and they were asked to 'identify' and 'name' the various constituents of the picture(s). After obtaining a list of 'terms', the experts were asked to 'define' each of them in their own words (natural language definition). Sometimes, by way of explanation, the experts provided a specific 'context' in which they discussed certain terms/concepts. The expert was then asked to group/categorise the terms in any way s/he felt pertinent to the task. 'Relations' (and thereby 'constraints') were often explicated, by asking the expert to explain the rationale or criteria behind each categorisation.

### 2.3.2 Step 2: Eliciting Instances

Next, stimuli, sets of visuals with screen-shots of typical applications that run on the PC, were shown to each expert. Upon identification of each application/ software, s/he was asked to provide a detailed specification of a suitable configuration for a PC that would run such applications. Typical applications considered were word processing, spreadsheets, database, e-mail, network access, Microsoft Windows 95, Windows NT, Linux, financial analysis, CAD/CAM, multimedia, demanding database applications, generic 32-bit applications, various programming environments, graphic-intensive applications, desktop publishing, and video conferencing. As a consequence of this exercise, we obtained several instances of expert-specific PC specifications.

### 2.3.3 Step 3: Validating the Conceptual Structures

Finally, the expert was shown several examples of specifications on a set of 'cards'; with one specification per card; and asked to 'sort' the cards in various groups/categories until all cards were sorted (Ch. 8: "Knowledge-Elicitation Techniques" in [11]). The expert was then asked to specify the feature/attribute, which had been used to determine the sort and the corresponding values for each of the piles. The expert was then invited to repeat, the sorting as many times as they wished – at each cycle they were asked to specify the feature and the corresponding values. After s/he had provided the features and their definitions, as discussed above, the expert was asked to organise them in any order s/he considered relevant. One of the experts, for instance, decided to give a 'level number' to each set of terms that he thought can be grouped together. For example, level '1' indicates entities that are external and visible, while level '2' designates all the components or devices that are inside the machine. The experts also assigned a 'context', to help clarify the perspective taken by them while defining a certain term.

## 2.4 Preliminary Analysis

We have been able to elicit the experts' conceptualisations of their domain (terms, definitions, concepts, relations). The experts also grouped the items in various categories and explained the rationale behind the ordering thereby validating their distinctive conceptualisation of the domain. This enabled us to create a hierarchy of levels, while evolving the ontology.

### 2.4.1 Glossary Building

A list of the domain terms and phrases used by each expert was compiled from her or his protocols and incorporated into a collective glossary. For each concept, suitable definitions and citations were extracted, and a cross-reference to the relevant protocols was made. Finally, standard definitions of essential terms taken from industry-recognised references [4] and [13] were incorporated alongside the experts' descriptions to aid comparisons. We were able to obtain a rich data set of around 600 domain concepts/terms.

## 2.5 Confirmatory Analysis and Construction of Ontologies

A detailed analysis of the artefacts produced in KA phases 1 & 2 was carried out in order to compare domain concepts and relations and look for discrepancies. First, we extracted an 'interesting' set of terms/concepts from the glossary, which were semantically rich, and modelled each concept in a semi-formal representation.

We decided to employ conceptual graphs (CGs) [14] as they can be used to illustrate unambiguously domain concepts and the associations between them. CGs are formally defined in an abstract syntax and this formalism can be represented in several different concrete notations, viz., in a graphical display form (DF), the formally defined conceptual graph interchange form (CGIF), and the compact, but readable linear form (LF). Each expert's terminology was classified into clusters of similar terms, and categorised as groups of key concepts and their variants. We then extracted, from the expert's protocols, chunks of text that contained a key concept or at least one of the variants. The chunks of text were then marked-up by underlining all domain concepts of interest.

Based on a protocol analysis procedure described in [1], we developed a protocol-coding scheme that enabled us to identify concepts and relationships. Two coders working independently were involved in the analysis to avoid bias and to provide verification. (See [7] for details of the procedure). We present below an instance of protocol coding. Extracts from the experts' protocols (P1 and P2) were marked-up to identify key concepts (and/or their variants):

**P1:** "The other user who comes along and has a specific requirement – maybe they will need a machine with a very large screen, maybe they will need a machine with more than the normal amount of disk-space or they are looking to have a different machine which is: very large screen, lots of disk space, the fastest processor out there, and in that case I then advise them what their options are, what their limitations are..."

**P2:** "The area where somebody wanted a machine that is out of the ordinary, was, a department who need machines to install large number of images and process the images and communicate with other people there doing the editing of the journal and processing of images that are involved in that. And so therefore they wanted the fastest processor that they could get, they wanted the largest graphics card that they could get, and they wanted sometimes, something like thirty-six (36) Giga Bytes of disk space, and would have preferred more."

Figure 1 shows the CG that was drawn to denote the relationships between key concepts inherent in P1 and P2.

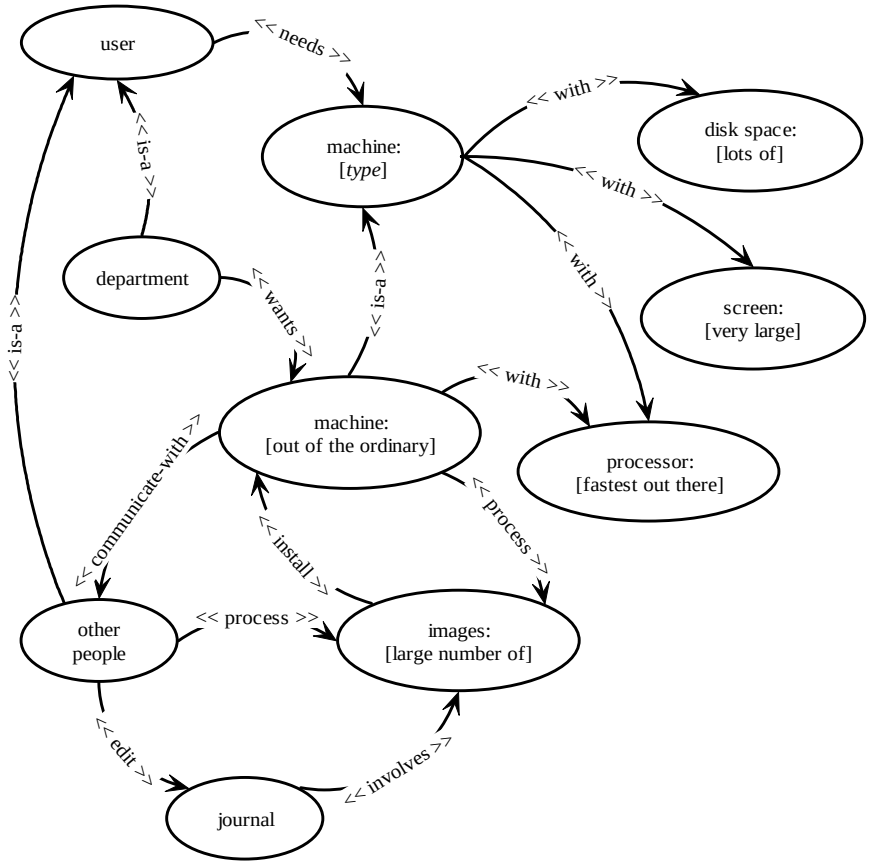


**Figure 1. Conceptual diagram illustrating a user requirement for an "out-of-the-ordinary machine" (refer protocol extracts P1 & P2)**

Following are examples of graphs written in linear form, representing fragments of conceptual relations in P2:

*"The area where somebody wanted a machine that is out of the ordinary, was, a department who need machines to install large number of images and process the images and communicate with other people…"*

```
a). [USER: Department] → (requires)
        → [MACHINE: out of the ordinary] -
              →1 - (install) -
              →2 - (process) -
                    → [IMAGES: large number of]
              →3 - (communicate-with)
                    → [USER: other people].
```

*"…other people there doing the editing of the journal and processing of images that are involved in that."*

```
b). [USER: other people] -
        →1 - (edit) → [JOURNAL]
        →2 - (process) → [IMAGES].

c). [JOURNAL] → (involves) → [IMAGES].
```

An organisation (hierarchical or otherwise) of terms, and their relationships is in essence an ontology that models the expert's distinctive conceptual structures.

# 3. Detecting Ontological Mismatches

Researchers in various areas of computing science are interested in automatic or tool-supported merging of ontologies (or class hierarchies, or object-oriented schemas, or database schemas – the specific terminology varies depending on the field). This has to be done regardless of whether the ultimate goal is to create a single coherent ontology that includes the information from all the sources (*merging*) or if the sources must be made consistent and coherent with one another but kept separately (*alignment*). Presently, the work of mapping, merging, or aligning ontologies is performed mostly by hand, but some tools are now being developed to automate the process, at least partially. However, both automatic merging of ontologies and creation of tools that would guide the user through the process and focus their attention on the likely points of action are in the early stages. Noy & Musen [10] provide a succinct overview of some of the existing approaches to merging and alignment in the field of ontology design, object-oriented programming, and heterogeneous databases. Klein [8] has analysed several problems that hinder the combined use of ontologies. Another recent work [9] uses articulation of ontologies to enable interoperation between knowledge sources.

## 3.1 Mismatches *among* Experts' Ontologies

While modelling the conceptual structures of each expert, we perceived that there were distinct differences among their individual ontologies. This was due to the fact that when knowledge was being elicited, it was not necessarily provided at the same level of abstraction or with the same measure of detail. Moreover, each expert has her/his own unique conceptualisation of the domain. In order to bring out these distinctions, we looked for discrepancies within and among their ontologies. Presented below are some examples of ontology mismatches in our domain of interest. We identified these by manual inspection, and related them to the three approaches introduced earlier in §1.1. It is interesting to note that discrepancies have been detected at an intra-ontology level as well as at the inter-ontology level. For reasons of space, we include only a few representative examples. Terms/ concepts in each expert's ontology are prefixed by the code '`e?-`', where '`?`' denotes a letter '`A`'..'`D`' to identify the expert who provided that particular term. For instance, '`eC-staff`' designates the concept '`staff`' in expert C's ontology.

Examples are represented here in a semi-formal notation only, as such descriptions allow easy comparison of definitions and conceptual relations.

### 3.1.1 Examples based on Visser *et al.'s* Classification [15]:

At the conceptualisation stage:

- Class mismatches
  - *Categorisation mismatch:*

    The following conceptualisations differ because the experts have partitioned the same class; as distinguished in their individual ontologies; into a distinct set of subclasses. The symbol '∪' signifies *union*, comprising the specified classes/concepts.

    `expert-C` *advises* `eC-staff ∪ eC-students`
    `expert-B` *advises* `eB-users ∪ eB-members-of-staff ∪ eB-departments ∪ eB-suppliers ∪ expert-A`

    `eC-PC` *is-made-of* `eC-parts ∪ eC-components`
    `eB-PC` *is-based-on* `eB-std-specs ∪ eB-std-options`

  - *Aggregation-level mismatch:*

    Here, the experts have identified the same (or similar) classes but defined them at dissimilar levels of abstraction. The symbol '→' denotes the relation '*is defined by*'.

    `eB-PC → eB-desktops ∪ eB-laptops`
    `eC-PC → eC-desktop ∪ eC-tower ∪ eC-portable ∪ eC-server`

    `eB-space → eB-disk ∪ eB-memory`
    `eC-memory → eC-RAM ∪ eC-VRAM`

- Relation mismatches
  - *Structure mismatch:*

    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.

    `eC-PC` *is-made-of* `eC-parts ∪ eC-components`
    `eB-PC` *has-component* `eB-processor`

  - *Attribute-assignment mismatch:*

    The ontologies differ here in the way they assign an attribute (class) to other classes. While expert B has assigned two disjoint attributes '`disk`' and '`space`' to the concept '`PC`', expert D defined a hierarchical relationship between similar concepts.

    `eB-PC` *has* `eB-disk`, `eB-PC` *has* `eB-space` versus `eD-PC` *has* `eD-disk` *has* `eD-space`

At the explication stage:

- *Concept & Term mismatch:*

  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.

  `eB-advice-to-users` ⇒ `eB` *advises* `eB-users` *about* `eB-spec`
  `eB-advice-to-supplier` ⇒ `eB` *advises* `eB-supplier` *about* `eB-spec`

  This is an instance of a discrepancy *within* the expert's ontology. It is imperative that such intra-ontology mismatches are resolved before any reconciliation with other ontologies is envisaged.

- *Concept & Definiens mismatch:*

  The symbol '←' denotes that the *definiens* which follow the arrow 'define' the term on the left-hand side of the description. The 'AND' operator is used here to concatenate multiple definiens. An incidence of multiple descriptions for the same term implies that the expert gave distinct definitions in different contexts.

  `eA-spec ← eA-supplier AND eA-standard-specs-list`

  `eB-spec ← eB-specifies AND eB-standard-specs-list`
  `eB-spec ← eB-specifies AND eB-machine`
  `eB-spec ← eB-description AND eB-machine AND eB-user`

  `eC-spec ← eC-requirement AND eC-user`
  `eC-spec ← eC-requirement AND eC-application`
  `eC-spec ← eC-specification AND eC-hardware-device`

- *Concept mismatch:*

  Identical terms and definiens, but each expert is referring to a quite different concept.

  `eB-min-spec ← eB-requirement AND eB-PC AND eB-user`
  *(referring to user's hardware requirement)*
  `eD-min-spec ← eD-requirement AND eD-PC AND eD-user`
  *(referring to system specification of software needed by user)*

- *Term & Definiens mismatch:*

  `eB-min-spec ← eB-processor-P3 AND eB-memory-128-MB`
  `eC-basic-PC ← eC-CPU-Pentium AND eC-RAM-64-MegaBytes`

  Although they use different terms and definiens, both experts are referring here 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.

- *Definiens mismatch:*

  Identical terms denoting same concept, but described by a varied set of definiens.

  `eB-fast-PC ← eB-processor-speed AND eB-memory-amount`
  `eC-fast-PC ← eC-CPU-Pentium3 AND eC-RAM-128-MegaBytes`

### 3.1.2 Types of Mismatch according to Wiederhold [16]:

*Key difference:*

`eB-RM-Mid-Range-System-Accelerator-Spec2` (reference for customer)
`eB-GCAT-03234` (reference for supplier & experts)

*Scope difference:*

`eB-advice` (advice given by eB to users, etc.)
`eA-advice` (technical advice sought by eA from eB)

`eA-memory`, `eD-memory` (referring to RAM)
`eB-memory`, `eC-memory` (referring to RAM and VRAM)

*Abstraction grain:*

`eC-faster-machine` (referring to speed of computer)
`eB-fastest-machine` (referring to speed of CPU)

*Domain semantics:*

`eB-budget` (funds/financial outlay available to user)
`eB-cost` (price of machine quoted by the supplier)

### 3.1.3 Examples to illustrate Shaw & Gaines' [12] four dimensions:

*Conflict:*

`eB-minimum-specification`: the given configuration of a standard specification
`eC-minimum-specification`: requirements of a certain hardware device
`eD-minimum-specification`: minimum system requirements for satisfactorily running a software

`eB-mid-range`: referring to the models of machines from one of the approved suppliers
`eC-mid-range`: the starting point from where he would evolve a customised configuration, given no cost constraints

`eA-requests`: "Unusual requests: e.g., for non-standard Ethernet card."
`eC-requests`: "…requests within the hard-disk."

`eA-specifications`: referring to suppliers' specification lists
`eB-specifications`: referring to (i) his role in specifying the machines; (ii) the standard specifications; (iii) a user's description of their old machine
`eC-specifications`: referring to (i) requirements of certain applications; (ii) user requirements; (iii) certain hardware devices (video cards & monitors)

*Correspondence:*

`eB-memory` versus `eC-RAM`
`eB-processor` versus `eC-CPU`

*Contrast:*

`eC-staff` versus `eB-approved-suppliers`

*Consensus:*

`eA-monitor, eB-monitor, eC-monitor` – all refer to video display unit/screen

# 4. Conclusion and Further Work

In this paper, we presented an approach to acquire knowledge and construct multiple experts' ontologies in a uniform way. We have shown how traditional knowledge elicitation and modelling techniques can be applied to the area of ontological engineering. Our work also highlights the need to reconcile mismatches among such ontologies, as a precursor to any distributed development, usage or management of multiple ontologies. A range of ontology mismatches was detected by human inspection, from simple syntactic inconsistencies to a rich array of semantic discrepancies, both at the conceptual and at a taxonomic level. We then compared our findings with relevant work in the areas of database interoperability, knowledge base reuse, and cognitive science.

We now aim to examine if these diverse approaches can be correlated and perhaps unified under a common framework. Subsequently, we intend to build KA/ontology revision tools that would automate some of this process.

# 5. Acknowledgements

We would like to express our gratitude to the domain experts. Thanks are also due to Ms. Caroline Green at the Department of Psychology for her assistance in the analysis and verification of protocols.

## References


[1] Alberdi, E., Sleeman, D., & Korpi, M. (2000). Accommodating Surprise and Taxonomic Tasks: The Role of Expertise. *Cognitive Science*, 24 (1), 53-91.

[2] Berners-Lee, T., Hendler, J, & Lassila, O. (2001). The Semantic Web. *Scientific American*, May 2001, 284 (5), 28-37.

[3] Clancey, W.J. (1985). Heuristic Classification. *Artificial Intelligence*, 27(3), 289-350.

[4] CMP Media, Inc. (2000). TechEncyclopedia. (source: *The Computer Desktop Encyclopedia*). http://www.techweb.com/encyclopedia/

[5] Ericsson, K.A., & Simon, H.A. (1984). *Protocol Analysis*, Cambridge, Massachusetts: MIT Press.

[6] Gruber, T.R. (1993). A Translational Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5, 199-220.

[7] Hameed, A., & Sleeman, D. (2000). Knowledge Elicitation to construct Ontologies in the domain of PC Specification. *AUCS/Technical Report TR0001*. Department of Computing Science, University of Aberdeen.

[8] Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. *Workshop on Ontologies and Information Sharing*, IJCAI'01, Seattle, USA, August 2001.

[9] Mitra, P., Kersten, M., & Wiederhold, G. (2000). Graph-Oriented Model for Articulation of Ontology Interdependencies. *Proc. of the 7th International Conf. on Extending Database Technology*, March 2000. Springer-Verlag.

[10] Noy, N.F. & Musen, M.A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Proc. of the 17th National Conf. on Artificial Intelligence* (AAAI-2000), Austin, USA.

[11] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., & Wielinga, B. (2000). Ch. 6: "Template Knowledge Models" and Ch. 8: "Knowledge-Elicitation Techniques" in *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, Massachusetts: MIT Press.

[12] Shaw, M.L.G., & Gaines, B.R. (1989). Comparing Conceptual Structures: Consensus, Conflict, Correspondence and Contrast. *Knowledge Acquisition*, 1(4), 341-363.

[13] Shnier, M. (1996). *Dictionary of PC Hardware and Data Communications Terms*, (1st Ed). O'Reilly & Associates. www.ora.com/reference/dictionary/

[14] Sowa, J.F., ed. (1998) Conceptual Graphs, *draft proposed American National Standard*, NCITS.T2/98-003.

[15] 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. *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA.

[16] Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. *IEEE Computer*, March 1992.

[17] Wiederhold, G. (1994). An Algebra for Ontology Composition. *Proc. of 1994 Monterey Workshop on Formal Methods*, September 1994.