In any modelling activity, a framework to determine the maturity of a developed model before its use is highly advantageous. Such a framework would save modellers expensive time in many areas of information systems. It would also lower the risk of users relying on an incomplete or inaccurate model. In this paper, we develop a framework which uses internal inconsistencies as a quantitative indicator for estimating the completeness and correctness of a model as it is cooperatively evolved. Whilst internal inconsistencies are due to bad fit between different parts of a model, we argue that they are also correlated with how the evolved model fits with the “world”. This argument underpins our framework to evaluate integrated models.

Contributions of this paper are three folds: firstly, it presents a theoretically grounded framework for integrating models. We extend an existing incremental modelling framework, NRDR, which represents multiple hierarchical restricted domains (MHRD), with automatic concept integration to allow NRDR to deal with multiple experts. Secondly, we couple this integration framework with a theoretically grounded monitoring process to assess the quality of the cooperatively developed model. Thirdly,
we illustrate an initial empirical study of our evaluation and integration framework in a computer hardware administration domain. We capture and integrate computer hardware models from several experts and we use our modelling evaluation framework to evaluate the resultant cooperative model.

Keywords: Modelling; model evaluation; cooperation; ontologies; knowledge acquisition; knowledge evaluation.

1. Introduction

In any modelling activity, a framework to determine the maturity of a developed model before its use is highly advantageous. Such a framework would save modellers expensive time in many areas of information systems. It would also lower the risk of users relying on an incomplete or inaccurate model. Examples where evaluating modelling would be of a significant economic advantage: in viewpoint-based requirement engineering in software engineering,\textsuperscript{10} in schema evolution of database systems,\textsuperscript{1–3} or whilst maintaining a knowledge based system using incremental development methods.\textsuperscript{9} In many instances, determining the maturity of an evolved model is done using external modelling expertise, e.g. Ref. 23 proposes a set of questions for “auditors” of a developed model in the context of developing corporate memories. Such approaches assume that on hand expertise outside that used in the modelling task is available. In this paper, we are concerned with developing maturity criteria in the absence of such sideline expertise. Our concern is similar to Refs. 21 and 22, but specifically for the case of cooperatively integrated models.

In this work, we argue that monitoring the integration process for internal inconsistencies between different versions or parts of an evolved model is an effective way to track the maturity of the model, and to ensure its accurate development. Our positive constructive view of inconsistencies is similar to Ref. 14. They point out that inconsistencies between requirement models may be desirable as they allow further elicitation (in capturing requirement models). Our constructive view is also similar to Refs. 21 and 22 who recommends tolerating some internal inconsistencies during model development. In what follows, we overview how researchers, from different areas of modelling, view inconsistencies within a developed model. We compare their views to our constructive view, which uses internal inconsistencies as a vehicle to formulate a maturity criteria of a developed model.

This paper is organized as follows: In Sec. 2, we discuss our work in the wider context of the literature and highlight its conceptual contributions. In Sec. 3, we present our modelling framework, and the theoretical basis for correlating internal consistency and completeness of a given model. In Sec. 4, we present our framework for cooperative modelling, describing how we integrate models for a given domain and how we merge common concepts during the integration process. In Sec. 5, as we illustrate our integration framework in an experiment in the domain of computer hardware administration, we also introduce notions to quantify the contribution of an individual modeler to the collaborative modelling effort. In Sec. 6, we use notions of Sec. 5 as basis of our monitoring process. This monitoring process is grounded...
in the abstractions of Sec. 3. Finally, we discuss our results and present directions our future work.

2. Related Work and Paper Contributions

In the ontology engineering community (a sub-community within the Knowledge Systems community), researchers have focused on creating and analyzing mechanisms to eliminate internal consistencies of models.\textsuperscript{9,27,28} They provided recipes to avoid inconsistencies occurring during the model development in different application areas.\textsuperscript{12,25} In the field of requirement engineering (RE), which involves eliciting, analyzing and resolution of ideas, multiple perspectives, and relationships at varying level of detail, the need to deal with multiple viewpoints is also central. We find that many approaches in modelling (in RE) avoid inconsistencies between models by disentangling any relationships which cause conflict. Easterbrook \textit{et al.}\textsuperscript{10} suggest to disassociate the modeler from the model, where each viewpoint is self consistent description of an area of knowledge with an identifiable originator. In this approach, the viewpoints do not correspond to people and they are not used to represent views of the world but together represent a single view of the many agents of the world. Similar to that, Nuseibeh \textit{et al.}\textsuperscript{24} define viewpoints to be loosely coupled, locally managed distributable objects encapsulating partial representation of knowledge about a system and its domain. They provide a framework that draws together organizational notions of an actor, knowledge source, role, or agent with the notion of a view held by the Easterbrook \textit{et al.} Nuseibeh’s framework facilitates the separation of concerns and structuring of software development knowledge.

Both works, of Easterbrook and of Nuseibeh, avoid the question of external consistency (maturity of the model) with the world. Other RE researchers also completely stultify this question by accepting that competing world views cannot be reconciled, and moreover that there is no such thing as a complete model which satisfies everyone, e.g. Refs. 13 and 31 espouse for managing inconsistencies rather than resolving them in requirement engineering. That is, they accept that the maturity of the model (the external inconsistency) cannot be objectively decided, as it is interpreted as simply a different conceptualization of the problem. A parallel view commonly held within the ontological engineering community is that to check the external consistency, between an ontology and the world is not possible. To avoid the question of such a check, many adopt the view that a collection of experts is more capable than a single expert to construct an externally consistent ontology (model) e.g. Refs. 18, 19 and 30. But again, with this view, at what stage in the integration process, a model should be considered mature, remains unanswered.

A more constructive view of external inconsistencies is taken in incremental development of knowledge systems. External inconsistencies with the world trigger discovery and resolution of internal inconsistencies.\textsuperscript{6} Following an incremental evolution in developing a knowledge model, some knowledge engineering and elicitation effort is transformed to maintenance effort: resolving internal inconsistencies is done
during maintenance of the knowledge system. That is, as inconsistencies with the outside world (errors) are detected, the system is updated and internal inconsistencies are resolved by a domain expert as the system evolves. External inconsistencies are detected as an incorrect behaviour which does not match a model of a domain’s expert behaviour. Our experience in the area of incremental modelling forms the basis of our approach, which can be summarized as follows: internal inconsistencies and external inconsistencies of a model are correlated; internal inconsistencies can be used to estimate the probability of external inconsistencies occurring. External inconsistencies are a measure of the maturity of a model.

We develop a framework to evaluate integrated versions of a domain model. We use internal inconsistencies to gauge the maturity of the evolved model. Our constructive use of local inconsistencies is similar to Ref. 14. The first step in our evaluation methodology is to detect (or estimate) the probability of internal inconsistencies. This estimate is then used to derive a trend of external inconsistencies. By monitoring internal inconsistencies, we ensure that a model generated is externally consistent. The intuition is that a coherent collective expertise is a better reflection of “reality”. That is, a state of consensus between experts reflects a mature model, which is consistent with the world as perceived by the collection of these experts.

We illustrate our evaluation methodology using an incremental modelling tool, NRDR, which we developed in Ref. 3. NRDR captured models are ontologies representing multiple hierarchical restricted domains (MHRD), similar to that employed by other authors (e.g. Ref. 11). Our integration approach follows our work in Ref. 29 which produced promising results in a number of domains. The integrated model/ontology is evolved as a result of a sequence of integration steps of model components from many experts, and resolving inconsistencies between them. We illustrate our approach with an example in a computer hardware administration domain. We capture and integrate computer hardware models (ontologies) from several experts. We use our modelling evaluation framework to evaluate the resultant community ontology.

3. Evaluating a Model Using Internal Inconsistencies

In this section, we overview the principles of our evaluation framework. We discuss how these principles align with the philosophy of incremental modelling. Our modelling evaluation framework is rooted in theoretical analysis of hierarchical incremental modelling. We adopt and generalize our incremental modelling approach from Ref. 5 to capture and integrate models from multiple sources of expertise. We first overview its technical details.

3.1. Incremental modelling

As earlier discussed in Sec. 1, two types of inconsistencies occur during modelling: the first type is internal inconsistencies, and are more likely where multiple sources
of expertise are used, or a modular approach involving integrating independently
developed modules is followed. The second type is external inconsistencies, which
relate to the choice of ontological units to represent the world, and how accurately
(and completely) the chosen model describes the world.

In the case of knowledge bases modelling, internal inconsistencies are usually
resolved during the development phase of a model, whilst resolving external inconsis-
tencies can be a longer term maintenance and testing task. This is true regardless
of the construction mechanism used, be it automatic such as in Refs. 17 and 19,
or relying on engineering approach such as KADS32 or ad hoc incremental such as
Ref. 15. To note though, in incremental knowledge construction of knowledge bases
(using incremental knowledge acquisition), maintenance of the KB and its develop-
ment are iterated into one process (e.g. Ref. 16). Incremental knowledge acquisition
restricts the change of the KB to the context which exposes the external inconsis-
tency. Only those parts of the KB which are lacking are identified and amended.
This in turn reduces the likelihood of internal inconsistencies. For our model eval-
uation purpose in this paper, our NRDR incremental modelling framework formally
shows that internal inconsistencies are correlated with external inconsistencies. In
what follows, we describe its technical details.

Experts use intermediate concepts that they (re) use in further explanations
when expressing their domain model. For example in chess, experts introduce
notions like “centre development” to justify some of their opening moves. When
asked to explain such intermediate concepts, experts often fail to provide a com-
plete explanation that always covers their use. Instead, they provide an operational
solution sufficient for the purpose of explaining the context on hand. Moreover,
experts articulation of intermediate concepts may depend on their articulation of
other concepts, which may not yet be made explicit or completely defined. NRDR
adapts the incremental KA process to match experts natural tendencies, in intro-
ducing explanatory intermediate concepts.

This enables experts to more easily express their knowledge, and to more effec-
tively build an operational KB. Semantics captured by an NRDR model are pow-
erful enough for most modern ontologies, NRDR models employ the most common
inter-conceptual relations. A model captured using the NRDR framework is an
ontology representing multiple hierarchical restricted domains (MHRD), similar to
that employed by other authors (e.g. Ref. 11). In particular, MHRD NRDR models
are sets of inter-related concepts with the following characteristics:

- They are defined through a set of attributes, so the presence of axioms between
these attributes is not considered. For example, in NRDR, domain instances are
represented as a feature vector of attributes.
- There can be taxonomic relations among the concepts, so that attribute (mul-
tiple) inheritance is permitted. For example, in NRDR, a model captures Is-a
relations between expert concepts.
• There can also be mereological relationships among concepts, i.e. part-of relationships are used. For example, in NRDR, rules can consist of concepts conjuncts to reflect part-of relationships.

Each NRDR concept is represented as a set of hierarchical rules with exceptions. Each rule is of the form, “If condition then conclusion”. This can be visualized as a tree (commonly known as Ripple Down Rules (RDR) tree.\(^7\)\(^8\) This representation of each model unit in NRDR allows experts to deal with exceptions and refine the definition of their units (or concepts) readily, without the need to modify or impact their past input. This way knowledge maintenance process is simple enough to be carried out by the expert himself. Definition of any term, \(X\), in NRDR starts with a default rule: “If true then default boolean conclusion about \(X\)” where the default conclusion is specified by the expert as \(X\) or its negation. Every term check starts at this rule. Every other rule can have two branches to other rules: a false-branch and a true-branch (exception-branch) (Fig. 3.1). The true branch of a rule is taken if its condition is satisfied, otherwise the false branch is taken. This is repeated until a terminal node \(t\) is reached. If the condition of \(t\) is satisfied then the conclusion of the rule in \(t\) is returned (\(X\) or the negation of \(X\)). If the condition of an exception rule (true-branch child rule) is satisfied it overrides the conclusion of its parent rule. If a false-branch leads to a terminal node \(t\), and the condition of \(t\) is not fulfilled, then the conclusion of the last rule satisfied on the path to \(t\) is returned for the concept definition. When the expert disagrees with the returned conclusion, he modifies the definition. An RDR tree concept is incrementally constructed by adding new leaf nodes when the expert wants to correct a failure. Rules are never deleted or modified. Child nodes are treated as exceptions to the parent rule, and each rule has only a local effect, which simplifies the modelling process.

Fig. 3.1. An RDR tree. Rules 1, 2, 3 are exceptions of rule 0.

\(^a\)Transitivity is not allowed for mereological organizations. Thus, the (mereological) part-of ontologies defined here will be supposed to hold irreflexivity (nothing is a (proper) part of itself), and asymmetry (if one thing is a proper part of another, then the second is not a proper part of the first).
Defined concepts can in turn be used as higher order attributes by the expert to define other concepts. This can represent a part-of or is-a relationship between concepts. A sub-model in the form of a concept hierarchy of an NRDR example is also shown in Fig. 3.2 (right) where $A$ is the highest level concept. An NRDR model requires modification if the expert disagrees with the conclusion returned by any of its RDR concept-trees. Two maintenance issues arise here: firstly, given a case that requires an NRDR model to be modified, the modification can occur in a number of places. The choice of refinement depends on the expert’s judgment. Secondly, localized updates in the hierarchical model can cause the expert to inadvertently introduce internal inconsistency to the emerging model. For example, if the expert modifies concept $D$ which is used in the definition of a higher concept $A$, this may cause past seen cases by $A$ to change. Following every model update, such internal inconsistencies are detected and fixed by the expert.

To construct and evaluate an ontology from multiple experts, we generalize our NRDR incremental modelling paradigm to include concept merging from different sources (this is described in Sec. 4.2). First we analyze the relation between internal consistency of a model and external consistency with the real world for our incremental modelling framework, NRDR. We show that as the internal inconsistencies of an NRDR developed model decrease, the accuracy of the KB, henceforth, the external consistency of the model increases.

### 3.2. Evaluation of incremental modelling

We showed elsewhere\(^5\) that the internal inconsistencies of an emerging NRDR decrease as the NRDR conceptual model becomes more accurate, i.e. as the model becomes externally consistent, it also becomes internally consistent. In this section, we overview that result and we argue its converse: as internal inconsistencies decrease, the external consistency (in the form of the accuracy of the concept definitions) increases. That is, as the model becomes internally consistent, it also becomes externally consistent. This result forms our evaluation framework for NRDR developed models later detailed in Sec. 6 (note that NRDR is generalized to multiple experts in Sec. 4 first).
We describe our results and observation in terms of the following definitions adapted from Refs. 4 and 26: given \( P(x) : D \rightarrow [0,1] \) the distribution function over the domain of expertise \( D \), and a rule \( r \) captured from an expert, we have the following notions:

**Definition 3.1.** The context of a \( r \), \( \text{context}(r) \subseteq D \) is the set of instances that get checked against \( r \).

**Definition 3.2.** The domain of a rule \( r \), \( \text{dom}(r) \subseteq \text{context}(r) \) is the set of instances which trigger \( r \).

**Definition 3.3.** The scope of a rule \( r \), \( \text{scope}(r) \) is the set of instances that trigger \( r \) and for which the conclusion of \( r \) is the correct classification.

**Definition 3.4.** The coverage of a \( r \), \( \text{coverage}(r) \) is the probability that a case \( x \), randomly drawn according to the distribution function \( P(x) \), is in \( \text{dom}(r) \). This is:

\[
\text{coverage}(r) = \sum_{x \in \text{dom}(r)} P(x).
\]

**Definition 3.5.** Predictivity measure \( \text{pred}(r) \) of a rule \( r \) is the ratio of all probabilities that objects in its domain are classified correctly. That is:

\[
\text{pred}(r) = \frac{\sum_{x \in \text{scope}(r)} P(x)}{\sum_{x \in \text{dom}(r)} P(x)}.
\]

The predictivity measure is a useful measure of the quality of the rules entered by the expert. In combination with the coverage of a rule, it reflects the utility of a rule. That is, how often a rule fires correctly for given number of cases. Note that the probability distribution of cases is taken into consideration, because the real performance of the conceptual model depends on instances with frequent occurrence, more than instances of rare occurrence.

An internal consistency takes place in an NRDR model, when a change of a concept \( C_1 \) causes an error in the definition of a concept \( C_2 \). This occurs when one or more of the following events occur: a case \( c \) is in the set of database of cases used to check the consistency of the model, \( c \) belongs to the domain of a newly added rule in \( C_1 \) and \( c \) at the same time belongs to the context of another rule in \( C_2 \). Hence, we can state, \( S_1 \), that: “As the probability of such an event occurring decreases, the coverage of newly added rules is decreasing.”

As the conceptual model develops, more cases are required to cause a new rule addition. As shown in Fig. 3.3, as the model develops, the coverage of a new rule decreases. Hence, we can state, \( S_2 \), that: “As the coverage decreases, the accuracy of the model is increasing.” (Fig. 3.3)

From \( S_1 \) and \( S_2 \), we can state that as: “As the probability internal inconsistencies decrease, the conceptual mode is also becoming externally consistent.” The
Fig. 3.3. Coverage of newly added rules as the conceptual model converges.

Fig. 3.4. Inconsistencies frequency versus NRDR model correctness: As the conceptual model gets larger, the incremental accuracy of the rules decreases. As the model size increases, the domain of the rules shrinks rapidly, taking down the probability of past cases becoming inconsistent close to 0. Hence, most of the inconsistencies occur in the early stages of developing an NRDR concept.

relationship between internal inconsistencies and external consistency of an NRDR model follows directly, shown in Fig. 3.4. In this paper, use NRDR as the basis of our incremental modelling because NRDR update mechanisms prove our earlier conjecture. The NRDR framework allows us to predict the likelihood of internal
inconsistencies, and this prediction is then used to assess the external consistency of a cooperatively developed model.

In this paper, we use our Object Oriented implementation of NRDR\(^3\) as an interface to model a computer hardware administration domain by many experts simultaneously. Each expert develops his own NRDR model, as these individual models are concurrently developed, integration steps are executed to merge the contribution of each model into a global model, implemented as a large NRDR model which incorporates all individuals contributions. Intermediate integration steps are automatically carried out. Following each integration step, experts gain access to each others’ intermediate representations and the evolving model. These steps integrate MHRD models obtained from the different sources (experts). Our approach to integrate these models consists of two components: a set of broad policies to manage and incorporate individual models (ontologies) into the large community ontology (model), and a set of lower level policies to integrate individual concepts from within individual models with existing concepts of the community ontology. Our approach is described in the next section starting with the high level policies.

4. Cooperative Modelling

In our approach, several model builders cooperate to construct an ontology (or a model).\(^b\) Models are represented by means of multiple hierarchical restricted domains (MHRD) (see previous section). In our framework, we can identify two types of users, namely, experts (model builders) and normal users (of the developed

\(^b\)Henceforth, we use the terms “domain ontology”, “ontology” and “model” interchangeably.
system). Experts do not need to be notified of changes made by each other, but each works independently so that his/her task is simplified. They can modify their model as soon as they have knowledge to append, modify or remove, without being blocked by other model builders. A global model is automatically generated from these individual models. Normal users can only query this global model, but they cannot make any amendment to it. In this section, we first describe the steps involved at the inter-model level, that is, how models are collectively handled. We then present its theoretical foundations showing how individual concepts (or terms) within the global merged model are handled.

4.1. Integrating models

Our integration framework addresses the following problems inherent to the cooperative work:

- **Redundant information**, that is, two different experts might attempt to describe the same part of the domain knowledge.
- **Use of synonymous terms for a concept**, that is, different experts can employ different terminologies for the same concept.
- **Inconsistent knowledge** which can be because some parts the model (ontology) are inconsistent with other of parts; or because two (or more) models are inconsistent with each other. As we will see later, tracking inconsistencies between models is the basis of the evaluation process which runs concurrent with the model evolution.

In our cooperative integration framework, the ontological knowledge pieces belong to the same application domain, so it is not necessary to take into account other possible linguistic issues such as homonymy, that deals with the problem of words that sounds in the same way but refer to different concepts. Moreover, we assume that all knowledge is transmitted via written media.

The integration process is carried out according to two basic principles:

- **User-dependency**: The integration process is to be guided by the knowledge supplied by the requesting (normal or expert) user.
- **Maximum information content**: The resulting global model is the one that provides the maximum information content, in terms of both quality (consistency and lack of redundancy) and quantity (number of entities).

Moreover, during the model integration process, there is also a terminological integration one. The different experts use their own terminology/vocabulary for assigning terms to their concepts. Hence, the global model must unify, as much as possible, these different terminologies. This process is also guided by the two abovementioned principles, in particular, they guide the selection of appropriate shared terminology in the process the integration of a set of models. In this process, the components of models are modified. Including a model in the integrated
The result of the integration process depends on these parameters:

- **The requesting user**: the integration principles are used to guide decisions made during the integration process. The user-dependency principle is applied if the requesting user possesses a model in the set of integrating models. It should be reminded here that only experts can possess models; otherwise, the maximum information content principle is applied.

The global model involves performing a vocabulary translation in order to adapt the terminology used. This translation is reflected in the axioms, so that those belonging to one model can be included in another one. It is a dynamic process, its operation depends on the parameters received. Details of this process is described in the next technical sub-section.

Fig. 4.1. The information/knowledge flow that occurs during the integration process: The user (either expert or normal) requests for the integration of a certain topic. This request is captured by an interface agent (the OO NRDR interface) which establishes the parameters of the integration process by querying the ontology base. Once the parameters are set up, the interface agent requests for integration to the integration agent, which performs the integration process by querying the ontology base. Finally, the integration results are sent by the integration agent to the requesting user via the interface agent.
• **Consistency between the models to be integrated**: this is detected by checking whether different models are inconsistent with each other. Two models are considered inconsistent if they have inconsistent definitions of, at least, one concept. The inconsistency may arise in different ways: two concepts might have the same name (term) but they do not have any common attributes (specific or inherited). Alternatively, they might have the same term and common attributes, but they do have a common parent or child concept. Such inconsistent models do not take part in the same integration process.

• **Redundancy in the set of models to be integrated**: this is measured in terms of equivalency. Two models are equivalent when every concept of each model has an equivalent concept in the other model. Two concepts are equivalent if their respective sets of attributes and parent/child concepts are equivalent.

• **The amount of knowledge contained in a model**: this is evaluated according to the maximum information criterion aforementioned. Different models are compared according to the number of concepts they have.

An important feature in our integration process is the use of a reference ontology. This is used as a guide for the elicitation of new models, and potential terminological transformation for existing ones. The selection of this reference ontology depends on who requested the integration process. In case the requester possesses a model in the integration process, their model will be the reference ontology. Otherwise, the model with the largest number of concepts is considered the best to use as a reference ontology.

Steps of the integration process (shown in Fig. 3.2) are: initialization, selection, instantiation and transformation. During initialization, models to be integrated are produced and a set of candidate models, \( C \), is kept. From \( C \), models that produce redundancies or inconsistencies are removed. For a pair of models to be integrated together, they cannot be inconsistent nor equivalent. Integrating inconsistent models would incorporate contradictory knowledge into the global model, whilst integrating equivalent models would incorporate redundancy into the
integration-derived model. Removal of inconsistent and redundant model yields the set of models to be integrated, \( I \).

The terminology of each ontology (model) in \( I \) is then transformed into a shared terminology. Details of this is shown in the next section. The terminology of the reference ontology is used as a guide. The models obtained from this terminology unification process form the so-called \textit{instantiated and integrated model}, \( G \). The reference ontology is taken as the skeleton of the temporary global model, \( G \), which is then used to generate the global model \( G_f \) as follows: the rest of models are sorted out according to their number of concepts and they are inserted into \( G \) in the order established by that sorting process. This minimizes the number of modifications required. The more models are included, the more knowledge the transformed model has. Through this process, new concepts, attributes and relationships are included to eventually produce \( G_f \).

\textbf{4.2. Managing concepts within the merged model}

Each model (ontology) is processed one concept at a time. If there are no other equivalent or synonymous concepts in the transformed model \( G \) (see previous section), the examined concept is added as it is, to \( G \) in its corresponding MHRD location. Otherwise, concepts, which are synonymous or equivalent, are first unified in terms of attributes and parent concepts before they are added to their corresponding MHRD location.

Generally, when domain models are developed by different experts, their conceptualization of the domain may vary resulting in different models. There are a number of reasons why different experts would develop different models. In the simplest case, experts merely use different names for the same concepts. However, it is rather common that different experts use in fact different domain conceptualizations. This is already evident from the fact that people develop different data models for the same task when it comes to database design.

The differences in different conceptualizations may range from merely different naming via superset and subset relations between “corresponding concepts” to entirely incommensurate models. Incommensurate models would not allow to compare individual concepts with each other because the models are based on incomparable data models. In that case the set of objects to which the models are to be applied are simply different and no simple superset/subset relation can be established. In the case of incommensurate models, no automatic way of relating them to each other appears possible. However, in the case that models refer to the same set (or subset) of objects, and once domain models have been sufficiently formalized so that an existing database of object descriptions can be automatically classified, it is possible to estimate the extensional difference of concepts.

The \textit{extension} of concepts is of interest as it reflects the practical usage of a concept rather than the theoretical or intended meaning. For example, the concept of an elephant may be defined in model \( A \) as \textit{a beast with a trunk} and in model \( B \)
as a grey animal weighing more than two tons. While the two definitions are quite different in their description of an elephant, if applied to a typical environment, they may apply to exactly the same set of objects. In contrast to that, the intension of the two concepts is quite different. This becomes only tangible, if there were objects that would be differently classified by the two concept definitions, e.g. a genetically modified dog with a trunk would be subsumed under the first definition but not under the second definition. The extension of concepts can be used to track such concepts defined in different models, and to judge whether they can or should be merged, either by a set-theoretic intersection or union. Such a merge operation can happen either with or without consultation of a domain expert.

**Definition 4.1.** Let $\text{Atts} = \{a_1, \ldots, a_n\}$ be a finite set of attributes, and let $r(a_i)$ be the range of attribute $a_i$. Then we call the Cartesian product of the ranges of all attributes the domain $D = r(a_1) \times \cdots \times r(a_n)$. Let $S$ be the set of objects in the object database on $D$. Let $c$ be a concept on the domain $D$ used for $S$. Then we call $c \cap S$ the extension of $c$ with respect to $S$.

**Definition 4.2.** Let $c$ be a concept on the domain $D$. Then we call $c \cap D$ the intension of $c$.

**Theorem 4.1.** Let $c_1$ be a concept in Ontology $O_1$ and $c_2$ a concept in Ontology $O_2$ which are said to correspond, i.e. they carry the same name in both ontologies. Let $c_m = c_1 \cap c_2$ be the concept resulting from the merging process of both ontologies $O_1$ and $O_2$. If $(c_m \cap (c_1 \cup c_2)) = \emptyset$ then, the extension of $c_m$ deviates from the target concept $c_t$ by at most $\varepsilon$ more than either of $c_1$ or $c_2$ with probability of at least $1 - \delta$, where the following inequality holds:

$$\delta < (1 - \varepsilon)^{1/k}$$

**Proof.** The condition $(c_m \cap (c_1 \cup c_2)) = \emptyset$ means that there is no object in the database for which the classification by $c_m$ differs from either $c_1$ or $c_2$. If $c_m$ deviates from the target concept by more than $\varepsilon$ than either $c_1$ or $c_2$, then the probability of drawing on object randomly according to $P$ falling into $(c_1 \cup c_2) \setminus c_m$ is greater than $\varepsilon$. If that is the case, then the probability of not having drawn any object falling into $(c_1 \cup c_2) \setminus c_m$ among $k$ randomly drawn objects is greater or equal to $(1 - \varepsilon)^k$. Since the database containing $k$ objects does not contain an object in $(c_m \cap (c_1 \cup c_2))$, we obtain $\delta < (1 - \varepsilon)^{1/k}$.

Theorem 4.1 allows to automatically merge two concept hierarchies that use the same name for corresponding concepts: for a required degree of accuracy $\varepsilon$ it bounds the probability that a merged concept definition will deviate from the target concept by more than $\varepsilon$ over the maximum deviation by any of the two merged models. That is, it allows to perform merge operations automatically, if the probability of an inaccurate result is acceptable. If the chance of obtaining an inaccurate merged
concept is unacceptably large, a human expert can be requested to approve a proposed concept merge operation. For the rest of the paper, we describe our evaluation framework of an integrated model. Towards this, we present details of our integration framework in an experiment in the computer hardware configuration domain. These results will then be analyzed, and followed by a statistical framework to evaluate an integrated model (ontology).

5. Experiment: Cooperative Modelling in Computer Hardware Configuration

In this section, we describe our model integration experimental workbench and the results obtained following two integration iterations. We first describe our experimental setup.

5.1. Experimental model integration setup for computer hardware domain

Our starting point is a set of 5th year students of Computer Science Engineering at the University of Murcia, who are registered in the Knowledge Acquisition module. The goal of the practical part of this module is the cooperative creation of a Computer Hardware maintenance model (ontology). Students are asked to act as experts in Computer Hardware and to each build individually a model of their expertise. Models (ontologies) are then integrated together into one large community ontology (Merged Model). Experts are then given access to this community model. They are independently asked again to modify their private model so that a second integration round was performed. The only external influence on the experts is the access to the integrated community model. In order to keep this experiment manageable, the experts are asked to limit the size of their models to 30 concepts. This will also force our experts to introspect and filter out their most important domain entities. This may result in very similar models if they describe the same part of the domain, or very different ones in case they focus on different parts. We will aim for an ontology of at size at least 100 concepts to ensure that experts introduce new concepts outside any overlap.

5.2. Integrating models

Table 1 shows the numeric description of the models built by the experts in this first step. Ontological elements contained in the models are concepts, attributes and relations (i.e. taxonomic and mereological). For evaluating the evolution of integrated model, only relations and concepts are taken into account.

The iterative monitoring process is now as follows: first, the models built by experts, $E1$ and $E2$, are integrated (see Sec. 4 for integrating concepts) together and results obtained are stored. At each following step, a new model is added to the integration process, so that, the effect a new ontology provoked to the integrated
Fig. 5.1. (Top): All cooperating experts are given access to the set of past seen instances in the domain. This is stored in a central object oriented database. (Bottom): In addition, experts can view the hierarchy of concepts, and update any concept in this hierarchy. In this section, the experiment performed for monitoring the process of integrating models is described, as well as the results obtained from it. Let us summarize the aforementioned assumptions for this experiment.
Table 1. Models prepared for the first integration step.

<table>
<thead>
<tr>
<th>Expert</th>
<th>Taxonomy</th>
<th>Mereology</th>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>24</td>
<td>17</td>
<td>43</td>
</tr>
<tr>
<td>E2</td>
<td>27</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>E3</td>
<td>23</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>E4</td>
<td>22</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>E5</td>
<td>20</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>E6</td>
<td>24</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>E7</td>
<td>49</td>
<td>16</td>
<td>68</td>
</tr>
</tbody>
</table>

Table 2. First integration step results.

<table>
<thead>
<tr>
<th>Expert</th>
<th>Relations</th>
<th>Concepts</th>
<th>Total Impact %</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>24</td>
<td>17</td>
<td>43</td>
</tr>
<tr>
<td>OINT E1-E2</td>
<td>51</td>
<td>21</td>
<td>70</td>
</tr>
<tr>
<td>OINT E1-E3</td>
<td>74</td>
<td>27</td>
<td>93</td>
</tr>
<tr>
<td>OINT E1-E4</td>
<td>95</td>
<td>32</td>
<td>116</td>
</tr>
<tr>
<td>OINT E1-E5</td>
<td>108</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OINT E1-E6</td>
<td>129</td>
<td>43</td>
<td>144</td>
</tr>
<tr>
<td>OINT E1-E7</td>
<td>171</td>
<td>54</td>
<td>181</td>
</tr>
</tbody>
</table>

ontology is analyzed and measured. This is repeated until the seven models are included into the integration process. The following measurements are taken at each step:

- Details of the source models (Table 1) and the integrated model: this category accounts for the number of concepts, taxonomic relations, and mereological relations contained in the models.
- Overlap between the incoming model ON and the integrated model OE, Overlap (OE, ON) = (X + Y) − Z, where X is the number of ontological entities in ON, Y is the number of ontological entities in OE before including ON, and Z is the number of models contained in OE after including the ON.
- Impact of the incoming model ON, Impact (ON, OE) = (Z − Y)/(X + Y). Z − Y represents the new ontological entities in OE due to ON. X + Y represents the ontological space in which the integration process takes place, comprised of the previous OE and the ON.
- Difference between the integrated model and the incoming model, Difference (OE, ON) = Z − Y.

Table 2 shows results obtained in this first integration step. Each row represents the integration of an incoming model with the set of already integrated ones. For each integrated model the table shows the following: the number of taxonomic relations, the number of mereological relations, the number of concepts and the number of ontological entities; the table also shows the impact on the integrated
ontology. In order to calculate the average impact each incoming model has on the integrated model, the impact value for the first model (E1) is not taken into account provided that not real integration is performed. The average impact for the first integration step is then $(39 + 27 + 19 + 11 + 11 + 18)/6 = 21\%$.

A second integration step is done, when this follows the experts’ modification of their original model in response to browsing the integrated model. Details of these new models are shown in Table 3. They are larger, in fact the overall number of ontological entities in this second set of models is 27\% higher than in the first set. A similar iterative integration activity is performed again. Results of this are shown in Table 4.

In the second iteration, the average impact on the integrated ontology is 16.5\% (Table 4).

The models impact has been so far evaluated by measuring the variation in the integrated model. However, it can also be obtained by considering the degree of support an expert’s model receives from the community. The methodology for calculating such degree of support can be described as follows. Let $O$ be the model obtained through the integration of $n$ models: $O_1, \ldots, O_n$. Then, the degree of support obtained by the model $O_i$, written $\text{degree of support}(O_i, O)$, is:

$$\text{degree of support}(O_i, O) = \text{relations support}(O_i, O) + \text{concepts support}(O_i, O).$$

$\text{concepts support}(O_i, O)$: For each concept $c$ in $O_i$, its score is given by the amount of experts that have also defined $c$ in their respective models. Moreover, each common

<table>
<thead>
<tr>
<th>Expert</th>
<th>Relations</th>
<th>Concepts</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Taxonomy</td>
<td>Mereology</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>25</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td>E2</td>
<td>36</td>
<td>8</td>
<td>45</td>
</tr>
<tr>
<td>E3</td>
<td>23</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>E4</td>
<td>40</td>
<td>8</td>
<td>44</td>
</tr>
<tr>
<td>E5</td>
<td>25</td>
<td>10</td>
<td>37</td>
</tr>
<tr>
<td>E6</td>
<td>39</td>
<td>6</td>
<td>44</td>
</tr>
<tr>
<td>E7</td>
<td>51</td>
<td>16</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expert</th>
<th>Relations</th>
<th>Concepts</th>
<th>Total</th>
<th>Impact %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Taxonomy</td>
<td>Mereology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>25</td>
<td>25</td>
<td>52</td>
<td>102</td>
</tr>
<tr>
<td>OINT E1-E2</td>
<td>58</td>
<td>32</td>
<td>73</td>
<td>163</td>
</tr>
<tr>
<td>OINT E1-E3</td>
<td>80</td>
<td>38</td>
<td>83</td>
<td>201</td>
</tr>
<tr>
<td>OINT E1-E4</td>
<td>117</td>
<td>44</td>
<td>97</td>
<td>258</td>
</tr>
<tr>
<td>OINT E1-E5</td>
<td>132</td>
<td>49</td>
<td>106</td>
<td>287</td>
</tr>
<tr>
<td>OINT E1-E6</td>
<td>164</td>
<td>54</td>
<td>132</td>
<td>350</td>
</tr>
<tr>
<td>OINT E1-E7</td>
<td>207</td>
<td>67</td>
<td>160</td>
<td>434</td>
</tr>
</tbody>
</table>
attribute adds an extra point to that concept. Thus, the score will be the sum of the score obtained by each concept.

relations\textsubscript{support} \((O_i, O)\): For each relation \(r\) in \(O_i\), its score is given by the amount of experts that have also defined \(r\) in their respective models. Thus, the score will be the sum of the score obtained by each relation.

In order to calculate the degree of support for each model, the global integrated model must be first obtained. Tables 5 and 6 contain the degree of support received by each expert’s model from the set of models that have been integrated. As it has been said before, the degree of support can be seen as a consensus measurement, since a higher score indicate that the knowledge contained in that ontology is supported by the rest of the users.

The average impact of the models on the integrated one is lower (using both methods in Table 6 or Table 4) in the second integration pass. This reinforces our initial hypothesis that the integrated ontology would change less and less as integration steps are carried out so that there will be a point at which changes to the integration ontology would not be significant, so that it could be said that consensus is reached.

Our integration framework provides results that will be analyzed by our evaluation framework. Our evaluation framework, is based on the analysis presented in Sec. 3, where we argued that internal inconsistencies, or clashes between components of a model, are a reflection of its incompleteness. In the next section, we

<table>
<thead>
<tr>
<th>Expert</th>
<th>Degree of Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>214</td>
</tr>
<tr>
<td>E2</td>
<td>176</td>
</tr>
<tr>
<td>E3</td>
<td>190</td>
</tr>
<tr>
<td>E4</td>
<td>144</td>
</tr>
<tr>
<td>E5</td>
<td>116</td>
</tr>
<tr>
<td>E6</td>
<td>203</td>
</tr>
<tr>
<td>E7</td>
<td>341</td>
</tr>
</tbody>
</table>

Table 5. Degree of support for the first integration step.

<table>
<thead>
<tr>
<th>Expert</th>
<th>Degree of Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>498</td>
</tr>
<tr>
<td>E2</td>
<td>444</td>
</tr>
<tr>
<td>E3</td>
<td>248</td>
</tr>
<tr>
<td>E4</td>
<td>300</td>
</tr>
<tr>
<td>E5</td>
<td>167</td>
</tr>
<tr>
<td>E6</td>
<td>352</td>
</tr>
<tr>
<td>E7</td>
<td>444</td>
</tr>
</tbody>
</table>

Table 6. Degree of support for the second integration step.
develop a statistical framework based on that principle; that the more changes are required to stabilize a cooperatively developed model, the less mature this model actually is. In other words, a state of convergence between experts is sought, and this state is assumed to reflect the world more accurately.

6. Evaluation of the Integrated Model and Conclusion

Earlier in Sec. 3, we presented the overall principle to evaluate an integrated ontology, which relates an integrated model completeness to its internal inconsistency. We proved this for our NRDR modelling technique. In this section, we complete the details of the evaluation framework, by providing a quantitative analysis to predict internal inconsistencies. Using the relationship between internal inconsistencies and completeness of a model (shown in Sec. 3), the quantitative prediction can then be mapped to a prediction about the completeness of a model. We first discuss our experimental results.

In our integration approach, models are developed by different and independent experts, and advanced computer science graduates. Models are then automatically integrated together. This integration process is monitored in order to observe whether consensus can be reached by using this incremental, cooperative way for developing models. For this purpose, we performed two integration steps. Comparing sets of models at the onset of each integration step, models belonging to the second set are larger than the ones included in the first one. However, the average impact is lower among the latter ones. This must be due to a higher overlapping among the respective models. The degree of overlapping among models shows how similar models are. Therefore, when moving towards consensus the overlapping must increase. Experts refine their models by using their own knowledge and the knowledge contained in the integrated model, which can be seen as knowledge accepted by the community, since the integrated model only contains (internally) consistent knowledge. In this manner, the integration framework integrates those knowledge pieces (i.e. concepts from individual models) that do not present consistency conflicts.

On the other hand, our hypothesis is also reinforced by the results shown in Tables 5 and 6. It can be noticed in those tables that models from the second integration step achieve higher degree of support than the ones from the first integration step. This is due to a higher overlapping in between the second integration step. Experts, when modifying their private ontology, are guided by the integrated ontology.

To evaluate the resultant integrated model in the presented experimental framework, there are two questions to be answered: first, how many models do we use. Second, how many iterations do we ask the experts to do. As we earlier discussed in the paper (see Sec. 2), testing a model against data is not always possible. Instead, we assume that when each expert has contributed as much as they can, the resultant model is effective (and correct). We formally articulate this view for the rest of
this section. This results in an outline of a statistical framework towards indicating when the integration process should be halted.

6.1. Evaluation of the integrated model

In Ref. 4, we showed how an incremental knowledge acquisition process can be evaluated without the need for a separate testing stage. This was achieved by a statistical estimation of key knowledge acquisition parameters coverage and predictivity of newly added knowledge (see Definitions 3.4 and 3.5 in Sec. 3), which estimate the likelihood that a false-positive or false-negative occurring as a result of a knowledge base (KB) amendment. This allows tracking the convergence of the KB. The strength of this approach is that no separate test data is required, which is the reason why we adopt the underlying ideas here for evaluating an integrated model. In the case of a KB, the performance of the KB during construction is monitored instead of a separate testing stage. For the process of integrating models, we monitor how likely it is that the next model will contribute to the integrated ontology. Adapting the KB monitoring paradigm to evaluating the integrated model is based on relating a case in the incremental KA process, to a model from a single expert in forming an integrated model. This gives us the likelihood of an internal inconsistency occurring as a result of integrating a single model. This in turn allows tracking the completeness of the integrated model using our results in Sec. 3 (Fig. 3.4).

In Ref. 4, the expert monitored how the KB classified a stream of data during the evolution of the KB by relying on the principle that the longer the KB can perform correctly without the need of the expert to intervene, the more accurate the KB is (by assumption). Data is an integral component of this monitoring process. Unlike a KB, an integrated model cannot be easily evaluated against data. To monitor the development of the integrated model, we monitor the potential contribution of each new model to the integrated model. In Sec. 5, we showed two ways to quantify this contribution, either by taking the impact value on the integrated model, or by taking the support value (support by other experts). To monitor the whole modelling process, we observe the trend of contribution of individual models. If a model's contribution falls below a certain threshold, the model is ignored. We attribute such small differences to special features of the conceptualization of the individual expert, rather than to inherent incompleteness of the integrated model. We consider how much change each new model causes in the emerging model. As the rate of change converges, we approach consensus between models from different experts. The integrated stored model this way comes to represent a consensus between models provided by different experts. It is this consensus, which we monitor and predict. In monitoring this emerging consensus model, we are not directly evaluating its performance. As we argued in Sec. 3, as this consensus is reached, that is as internal inconsistencies disappear, the model is generalized and becoming complete.
In the following, we assume an arbitrary, fixed but unknown probability distribution $P_{ont}$, according to which we assume models provided by different experts are randomly drawn. This probabilistic model does not impose any restriction, other than requiring that the probability distribution does not change over time. A plausible requirement for a model of merging multiple models should require that the order in which the models are integrated should not have a major effect on the outcome of the integration process. The beauty of the following theorem comes from the fact that it involves an arbitrary function $\text{Difference}(\cdot, \cdot)$ that quantifies the difference between two models. There is no need here to be specific about that function and depending on the exact objective one can choose this function as one wishes. This function can be the degree of support or the degree of impact (of Sec. 5).

**Theorem 6.1.** We assume there are multiple models from different experts to be integrated. Given a threshold tolerance $\varepsilon$ for how different a model can be before it contributes to the existing model $OE$, after $m$ models randomly drawn from $P_{ont}$, each falling below the given threshold $\varepsilon$. Then the following inequality holds for the probability that the threshold $\varepsilon$ is not exceeded by the next model $ON$, randomly drawn according to $P_{ont}$, with probability of at least $1 - \delta$: $P(\text{Difference}(ON, OE) \leq \varepsilon) > (1 - \delta)^{1/m}$

**Proof.** We estimate the fraction of models in our pool that would exceed the threshold $\varepsilon$. Let us denote the fraction of models exceeding the threshold by $1 - p$ and accordingly the fraction of models not exceeding $\varepsilon$ by $p$. Then the probability that we draw $m$ models in a row none of them exceeding the threshold $\varepsilon$ is given by $p^m$. We require this probability to be at least $1 - \delta$, that is, we obtain the following inequality:

\[
p^m \geq 1 - \delta
\]

\[
\implies p \geq (1 - \delta)^{1/m}.
\]

Hence, there is a chance of at least $p \geq (1 - \delta)^{1/m}$ to encounter a model not exceeding the threshold $\varepsilon$ at the next random draw of a new model according to $P_{ont}$. Thus, Theorem 6.1 follows.

In the experimental setup of the previous section, a model formed by an expert after revisiting the community integrated model is less likely to contribute to future integration steps. This was evident in the two iterations. With respect to Theorem 4.1, newly prepared models by experts are considered independent. This way a finite number of experts can potentially generate an infinite set of models. For our statistical framework of Theorem 6.1, a function that estimates the difference between an incoming ontology ($ON$) and the integrated one ($OE$), $\text{Difference}(ON, OE)$ is necessary. Let $O_1, \ldots, O_n$ be the set of ontologies included in $OE$ and let $ON$ be the incoming ontology. Then, the difference between $OE$ and $ON$, written $\text{Difference}(OE, ON)$, can be, for example, be defined as follows: $\text{Difference}$
\( (OE, ON) = K(ON) - \bigcup_i (\text{overlap } (ON, O_i)) \), for \( i = 1, \ldots, n \), where \( K(ON) \) is the number of ontological entities in \( ON \); \( \text{overlap } (ON, O_i) \) is the set-theoretic intersection of the set of ontological entities in \( ON \) and in \( O_i \) respectively, for each \( O_i \) included in \( OE \).

7. Summary and Conclusion

This paper proposes coupling the collaborative construction of models with a statistical monitoring process, which oversees the contribution of each model, and indicates when to halt the cooperative modelling effort.

Specifically, the collaborative construction is stopped when the integrated model becomes stable. The collaborative construction is complemented with an (automatic) model integration process. Our work is based on the idea that internal inconsistencies in a cooperative modelling process are mainly a signal of incompleteness of the evolved model. This idea is theoretically grounded for our modelling framework, NRDR.

We showed how it is possible to monitor the (automatic) process of integrating multiple models originating from different experts. We also showed how to evaluate the contribution of an individual expert towards a consensus model. In essence, we statistically estimate how likely it is that the model presented by an expert will require a change in the integrated collective model, given a certain error tolerance level of \( \varepsilon \). We do this based on a probabilistic model of how the different models might relate to each other. Our Theorem 6.1 in Sec. 6 is based on the assumption that there the probability to obtain a certain ontology from the “new” expert does not change over time. Using Theorem 6.1, we are in a position to decide when to stop looking at more models from other experts. This saves knowledge managers and project developers on experts expensive time. While Theorem 6.1 addresses both, internal as well as external inconsistencies, depending on what difference function one chooses to use, we catered for internal consistencies and its impact on completeness and accuracy of a model in Theorem 4.1 in Sec. 4.

Our iterative integration framework (see Sec. 4) ensures that we converge to a single model with which every expert would be happy. The probabilistic model proposed in Sec. 4 allows to assess the empirical impact of integrating two different concept definitions stemming from two different models. This is simply because experts look at the integrated community model and see themselves in a wider community to which they contribute. This encourages consensus and circumvents difficulties due to possibly significantly varying conceptualizations of a domain. Even outside this iterative approach, we can estimate when we reached an integrated model with which a large fraction of experts will reasonably agree.

Our empirical measure for the difference between two concepts stemming from two different models (domain conceptualizations) is based on the intuitive notion of the probability that a randomly drawn object, according to some arbitrary fixed but unknown probability distribution, would be differently categorized by different
models. This was not necessary for our theoretical considerations in Sec. 6, as the probabilistic assumptions in Theorem 6.1 did not depend on that. Further work is required to develop a more detailed framework that would allow to distinguish between categorizing objects into strongly related and less related categories. More research is also required to cater for different levels of expertise or for priorities, which might be given to different conceptualizations.

Our initial results with the hardware administration is evidence for the practicality of the approach. As we monitored the integration process in our domain, and as the probability of the next model making a significant contribution decreased, we learnt that the integrated model has become stable. This is an indication that most domain experts would not change it. This is more likely to be reached quicker in stable well-established domains like ours, where large discrepancies between conceptualizations are less likely. In the case of incommensurate models, a number of problems would occur. Firstly, with respect to merging concept (as described in Sec. 4), no automatic way of relating concepts with wild discrepancies between them to each other appears possible. Moreover, the convergence stipulated in Theorem 6.1 and in Sec. 3, and observed experimentally in Sec. 5, would not be likely to occur.

To develop our work in less structured domains, and to accommodate different levels of expertise, we plan to make available our NRDR/OO system on the internet, and to pursue developing semantic web applications. A current workbench application under development for our approach is in e-learning. This should give us insights on how to handle concepts with wild discrepancies, and to test our approach in domains where no simple superset/subset relation can be established.

Acknowledgment

This paper has been possible thanks partly to the financial support of the Séneca Foundation, Coordination Centre for Research, through the Séneca Program (FPI). The research has been performed under projects FIT-150200-2001-320, FIT-070000-2001-785, and Seneca PL/3/FS/00. This work has also been supported by the American University of Beirut Research Board, under grants DDF 111130 088173 and DDF 111130 088126.

References


