Reconciling requirements: a method for managing interference, inconsistency and conflict

George Spanoudakis and Anthony Finkelstein

Department of Computer Science, City University, Northampton Square, London EC1V 0HB, UK
E-mail: {gespan,acwf}@cs.city.ac.uk

This paper outlines a method, called reconciliation, for managing interference between partial specifications or viewpoints. The method supports the detection, verification and tracking of ontological overlaps. The paper describes the heuristics on which the method is based and illustrates the application of the method using a scenario.

1. Introduction

The construction of a complex software system involves many agents (aka participants or actors). These agents have different perspectives or views of the artifact or system they are trying to describe or model. This gives rise to many partial specifications (or viewpoints) reflecting those perspectives [Nuseibeh *et al*. 1994; Maiden *et al*. 1995]. These specifications "interfere" with each other to the extent they refer to, or assert properties of common aspects of the system under development and its domain. This is a particular feature of the requirements engineering setting.

Interference between specifications can occur at two different levels. First, they might "ontologically overlap", that is they might incorporate components referring to common aspects of the "real world". Second, in cases where they overlap ontologically, they might also be inconsistent with each other. We believe that both ontological overlap and inconsistency are inevitable and acceptable in system development [Finkelstein *et al*. 1994]. Ontological overlap because it is necessary to support multiple perspectives, and inconsistency because it is a necessary to support innovative thinking, deferment of commitments and exploration of alternatives.

The consequence of this stance is that interference between specifications needs to be "managed", involving cooperation between the "viewpoint owners" [Finkelstein *et al*. 1994]. This is complicated by specifications which: use different languages; are at different degrees of abstraction, granularity and formality; deploy different terminologies; are at different stages of development. These complexities, set alongside the normal software engineering problems of scale, suggest the need for automated reasoning and method support for interference management.

The interference management that is required (or indeed possible) varies. In certain cases it might be loose, that is simply identifying ontological overlaps and

inconsistencies, in other cases it might be tight, involving the integration of the specifications and the resolution of their inconsistencies.

Reconciliation, the method discussed in this paper lies between loose and tight interference management. It supports the detection, verification and tracking of ontological overlaps. This is in some senses an easier problem than dealing with inconsistencies. It is amenable to the application of heuristic techniques, for example, inexact-matching mechanisms [Spanoudakis and Constantopoulos 1995] and models of computer-supported negotiation [Easterbrook 1991], while inconsistency detection may require theorem proving and sophisticated formal frameworks [Hunter and Nuseibeh 1995]. In any case the detection of ontological overlaps is prerequisite for detecting inconsistencies.

In the rest of the paper, we describe a heuristic method for reconciling viewpoints (section 2), we detail the heuristics on which it is based (section 3), give a scenario showing how it is used (section 4), briefly describe tool support for specification matching (section 5), review related work (section 6), and conclude with a discussion of open research issues and future work (section 7). An Appendix giving basic definitions is attached.

2. Overview

The method we adopt for reconciling viewpoints has two basic stages, namely *analysis* and *revision*, as shown in Fig. 1. It detects ontological overlaps using a computational model of similarity and a classification of specification components with respect to a meta-model of domain-independent, semantic modelling properties – analysis. It also supports the re-modelling of specification components so that the results of similarity analysis and viewpoint owners assessment of overlaps converge – revision. The goal of this process is to ensure that the modelling of specifications is consistent with the human assessment of ontological overlaps between them and establish a shared understanding among specification owners of the potential for inconsistency.

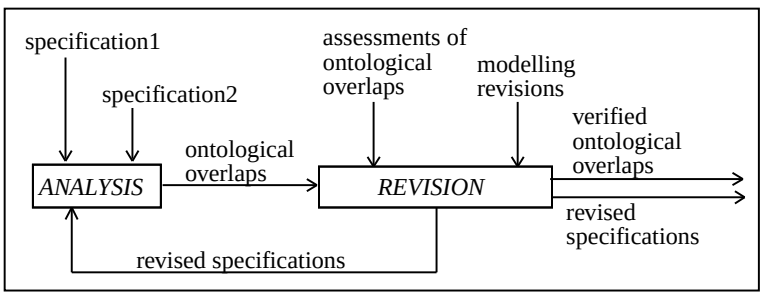

Figure 1. Reconciliation of viewpoints.

2.1. Analysis of specifications

The analysis of specifications is performed by a computational model of similarity. A specification is treated as an aggregation of "specification components", classified using a meta-model, which expresses general, domain-independent, semantic modelling properties. Both this meta-model and the specifications are described in Telos, an object-oriented knowledge representation language, supporting the semantic modelling abstractions of classification, generalisation and attribution [Mylopoulos *et al.* 1990].

2.1.1. Meta-model

The meta-model consists of a kernel and a set of extensions. The kernel provides the key, domain-independent, properties of semantic modelling schemes [Storey 1993; Motschnig-Pitrik 1993], whereas the extensions provide additional properties for modelling established specification languages.

The kernel part is organised as a generalisation taxonomy of classes, with specification components classified by the properties they possess. In particular, components are distinguished into those representing entities and those representing relations. Entity representing components are further specialized into natural, nominal, place, event, activity, state, agent and physical quantity components. Components representing relations are initially distinguished by their arity (for example binary or n-ary relations). Binary relations are further specialized according to: cardinality constraints (for example, $1:1$, $N:M$, total and onto relations); mathematical properties of relations (for example, symmetric, transitive and set-inclusion relations); existential dependencies between related items or other constraints between them, including their temporal coexistence, physical separability and substance homogeneity (Storey 1993).

The extensions to the meta-model comprise classes of additional properties for modelling established specification languages. Current extensions include constructs to support the description of specifications developed in the relational (for example fields, relations, inclusion dependencies) and object-oriented data models (for example object types, attributes and identifiers, is-a relations).

In essence, the meta-model enables the enrichment of the semantic content of specification components by asserting domain-independent properties about them and supports their representation with respect to a common set of structuring constructs. Both are prerequisites for the computational detection of their similarity. A detailed description of the meta-model is given in [Spanoudakis and Constantopoulos 1995].

2.1.2. Computational model of similarity

Specifications, described as Telos objects in terms of the meta-model, are compared using a computational model of similarity [Spanoudakis and Constantopoulos 1995; 1996; Spanoudakis 1994]. Similarity analysis is based on three metric functions, namely the classification, generalisation and attribution metrics, which measure

conceptual distances between specifications with respect to the classification, generalisation relations and the attributes constituting their descriptions.

The classification distance between specification components indicates their differences with respect to the properties expressed by the relevant classes of the meta-model. It is computed by identifying the non-common classes of two components, estimating the importance of these classes, and aggregating the importance measures obtained into a classification distance measure (function d_c in the Appendix). The generalisation distance reveals semantic differences between specification components, indicated by their non-common superclasses and is computed in a similar manner to the classification distance (function d_g in the Appendix). The attribution distance between specifications determines an optimal isomorphism I_s between their structures. Specification components contained in these structures are mapped only if they are classified under the same classes of the meta-model – *semantic homogeneity*. In cases where they can be mapped in many ways, the model selects the mapping with the minimum total distance (*minimum distance isomorphism* and the function d_a in the Appendix). The estimation of the pairwise distances between specification components uses recursive generation of isomorphic mappings between their own substructures.

The similarity analysis of two specifications results in:

(i) their classification, generalisation, attribution and overall distance measures;

(ii) a graph isomorphically mapping semantically homogeneous components at the successive levels of the structural closures of the specifications (the arcs of this graph are weighted by the pairwise distances of the mapped components);

(iii) a list with their common and non common classes, each weighted by its importance; and

(iv) a list with their common and non common superclasses, each weighted by its importance.

2.2. Revision of specifications

The isomorphism I_s between the components of two specifications is likely to reflect their ontological overlaps. However, flaws, incompleteness or lack of an adequate semantics in the modelling of specifications might force similarity analysis to generate mappings, which are incorrect. In such cases, specification owners can propose a different isomorphism I_o between specification components, which in their opinion correctly reflects these overlaps.

Our method uses the assessment of similarity matching by specification owners to suggest heuristic checks on, and subsequently revisions to specifications which would make I_s and I_o converge. The criteria forced similarity analysis to generate incorrect mappings between components, namely the *semantic homogeneity* and the *minimum distance isomorphism*, can be used to trace those mappings back to specific elements in specifications modelling and suggest revisions. Some of these revisions

resolve forms of inconsistency between specifications arising from incompatible classifications of ontologically coincident components under the meta-model. Other complete specifications with respect to each other. The method supports iterative revisions (Fig. 1) up to a point where I_s and I_o coincide completely. Revision stops at this point since the results of similarity analysis are confirmed as a correct reflection of the ontological overlaps between specifications. Along the way, the method guides specification owners through a disciplined check on the correctness and completeness of their specifications as well as systematic modification in line with their indication of the ontological overlaps between them.

3. Heuristics

3.1. Appraisal

Specification owners "appraise" the result of similarity analysis by suggesting an alternative isomorphism I_o between the ontologically coincident components of their specifications. In general, I_s and I_o will partially coincide. Components with and without counterparts in I_s will be referred to as *corresponding* and *unique*, respectively. Components mapped identically or left without any counterparts by both I_s and I_o will be said to be correct corresponding and correct unique components, respectively. Components with non identical mappings in I_s and I_o will be said to be wrong unique or wrong corresponding components. The characterization "wrong" for corresponding components means that their mappings by I_s do not correctly indicate ontological overlaps. Similarly, the characterization "wrong" for unique components in some specification means that, despite the absence of any counterparts for them in I_s, they should be treated as ontologically coincident with components of the other specification with which it is being compared.

We can further distinguish wrong components as:

(i) *Wrong unique components of type* 1 (*WU*1-*components*). These are unique components of one specification, which should have been mapped onto unique components of the other (according to I_o) although they have not by I_s (e.g., components $x1$ and $y1$ in Case 1 of Fig. 2).

(ii) *Wrong unique components of type* 2 (*WU*2-*components*). These are unique components of one specification that should have been mapped onto components of the other (according to I_o), which have been mapped onto different counterparts by I_s (e.g., component $x1$ in Case 2 of Fig. 2).

(iii) *Wrong unique components of type* 3 (*WU*3-*components*). These are unique components of one specification that should have been mapped onto components of the other (according to I_o), which did not exist at the time of comparison (e.g., component $x1$ in Case 3 of Fig. 2).

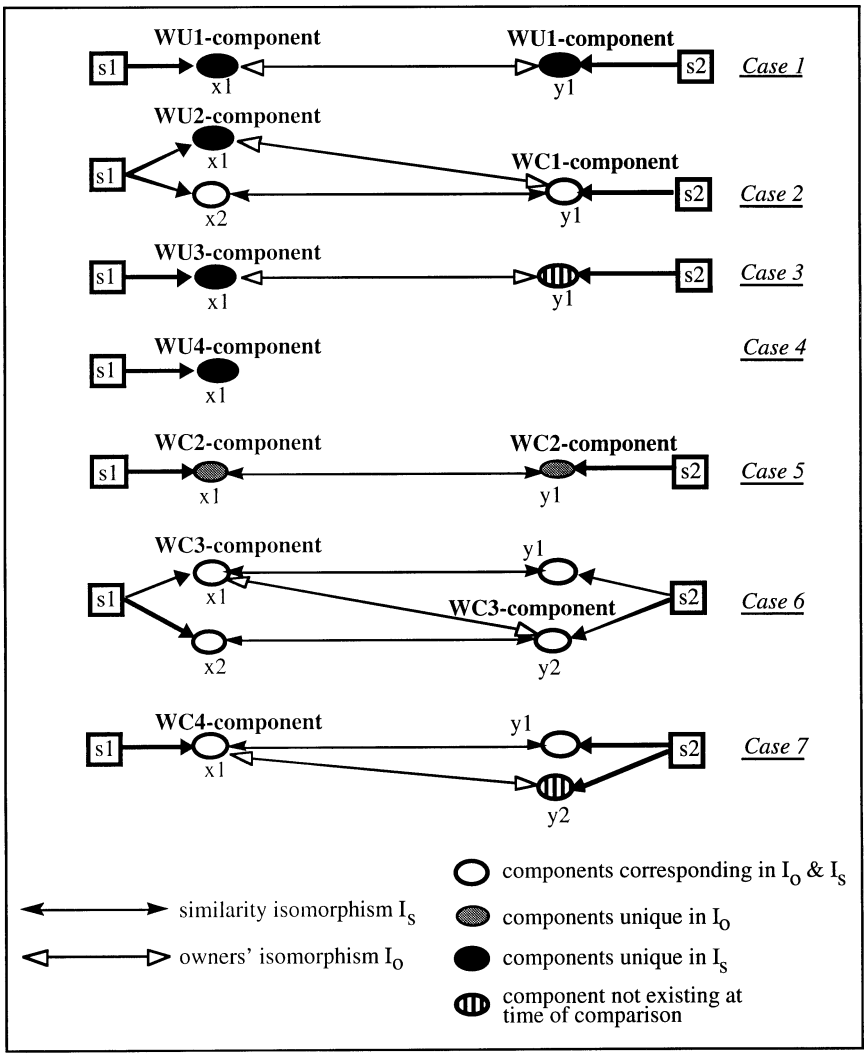

Figure 2. Wrong unique and corresponding specification components.

(iv) *Wrong unique components of type* 4 (*WU*4*-components*). These are unique components of one specification which are identified as redundant after the comparison with another specification fails to map them onto counterparts (e.g., component $x1$ in Case 4 of Fig. 2).

(v) *Wrong corresponding components of type* 1 (*WC*1*-components*). These are components which should have been mapped (according to I_o) on counterparts different from the ones which I_s maps them on and which have not been mapped by I_s (e.g., component $y1$ in Case 2 of Fig. 2).

(vi) *Wrong corresponding components of type* 2 (*WC*2*-components*). These are

components, which should not have been mapped onto any counterparts according to I_o although they have by I_s (e.g., components $x1$ and $y1$ in Case 5 of Fig. 2).

(vii) *Wrong corresponding components of type* 3 (*WC*3*-components*). These are components, which should have been mapped (according to I_o) onto counterparts different from the ones they have in I_s. Their appropriate counterparts have themselves been mapped onto wrong components by I_s (e.g., components $x1$ and $y2$ in Case 6 of Fig. 2).

(viii) *Wrong corresponding components of type* 4 (*WC*4*-components*). These are components, that should have been mapped (according to I_o) onto counterparts different from the ones they have in I_s, which did not exist at the time of comparison (e.g., component $x1$ in Case 7 of Fig. 2).

Examples of these cases are given in section 4, below. Based on these distinctions, our method provides a set of heuristics that can be deployed for tracing disparities between I_o and I_s back to the modelling of specification components and revising them so that I_o and I_s converge. These heuristics are discussed below.

3.2. Dealing with WU1-components

WU1-components may appear if similarity analysis does not map them because they were not classified under exactly the same classes of the meta-model (due to *semantic homogeneity*). Notice that, ontologically overlapping components should be identically classified because they are expected to share the same general semantic properties. For instance, it would not be reasonable to say that two components, classified as $1:N$ and $M:N$ relations respectively, express the same relationship in the real world. Therefore, classification discrepancies between ontologically overlapping components might be reasonably attributed to incorrect and/or incomplete classification with respect to the meta-model. Consequently, they need to be checked and possibly revised thereby enabling similarity analysis to generate the desired mappings. The following heuristics (expressed for the WU1-components of Case 1 in Fig. 2) may be applied in such cases:

H1: *Check if the non-common classes of* $x1$ *and* $y1$ *are correct and if not remove them.*

H2: *Check if any of the non-common classes of* $x1$ *and* $y1$ *should be classes of the other as well and add the relevant classifications.*

3.3. Dealing with WU2-components

WU2-components may appear due to the criteria of semantic homogeneity or the minimum distance isomorphism in similarity analysis. Because of the semantic homogeneity criterion, a WU2-component might not be mapped onto its desired counterpart

as a result of non identical classification. Such cases may be explored and revised in accordance with the heuristics H1 and H2, as in the case of WU1-components. In other cases, the mapping of the desired counterpart of a WU2-component onto a different, but wrong, component of the same specification (component $x2$ of Case 2 in Fig. 2) might be the result of an accidental incorrect common classification of them. Such cases can be explored and rectified using the following heuristics (expressed for the components of Case 2 in Fig. 2):

H3: *Check if the common classes of $x2$ and $y1$ are correct and if not remove them.*
H4: *Check if $x2$ and $y1$ should have been classified under any non-common classes of the meta-model, although they have not, and if so add the relevant classifications.*

Notice that H4 supports the elicitation of new information about the components involved.

If the classification checks do not resolve the problem, the desired mapping might be achieved by exploring why $d(x2, y1)$ is less than $d(x1, y1)$ and revising the modelling of $x1$ and $y1$ in order to reverse this inequality. Viewpoint owners need to consider specific aspects in the modelling of $x1$ and $y1$, which affected the partial conceptual distances between them and consequently their overall distance. Given that H1 and H2 have been applied revealing no classification discrepancies between the components (this implies that their classification distance equals 0), the overall distance inequality might be the result of inequalities between the generalisation and/or the attribution distances of the involved components (i.e., $d_g(x1, y1) > d_g(x2, y1)$ and/or $d_a(x1, y1) > d_a(x2, y1)$). Reversing any of these inequalities by revising the modelling of $x1$ and $y1$ can force similarity analysis map them onto each other. Below, we present heuristics guiding such revisions (expressed for the components of Case 2 in Fig. 2).

3.3.1. Reversing the inequality between the generalisation distances

Viewpoint owners may consider revising the generalisations of $x1$ to decrease $d_g(x1, y1)$ and the generalisations of $y1$ to decrease $d_g(x1, y1)$ or increase $d_g(x2, y1)$ or both.

(i) decrease $d_g(x1, y1)$. The following heuristics might be applied:

H5: *Check if $x1$ has been incorrectly generalised to its unique superclasses with respect to $y1$ and if so remove the relevant generalisations.*
H6: *Check if any of the unique superclasses of $x1$ with respect to $y1$, which are not superclasses of $x2$, should be superclasses of $y1$ and if so add the relevant generalisations.*

Notice that adding to the superclasses of $y1$ the unique superclasses of $x1$ which are superclasses of $x2$, would not affect the inequality $d_g(x1, y1) > d_g(x2, y1)$ since it would decrease equally both $d_g(x1, y1)$ and $d_g(x2, y1)$.

H7: *Check if $y1$ has been incorrectly generalised to its unique superclasses with respect to $x1$ but not $x2$ and remove the relevant generalisations if so.*

Notice that removing the unique superclasses of $y1$ with respect to both $x1$ and $x2$ would not affect the inequality $d_g(x1, y1) > d_g(x2, y1)$ since it would decrease equally both $d_g(x1, y1)$ and $d_g(x2, y1)$.

(ii) increase $d_g(x2, y1)$**.** The following heuristic might be applied:

H8: *Check if $y1$ has been incorrectly generalised into its common superclasses with $x2$, which are not superclasses of $x1$, and if so remove the relevant generalisations.*

Each non common superclass increases the generalisation distances between components by the inverse of its specialisation depth in the generalisation taxonomies of which it is a part. Therefore, it is possible to present viewpoint owners with the relevant sets of classes, which are to be added or deleted, ordered in ascending specialisation depths and identify those whose atomic modification could reverse the inequality $d_g(x1, y1) > d_g(x2, y1)$.

3.3.2. Reversing the inequality between the attribution distances

The attribution distances between $x1$ and $y1$ and between $x2$ and $y1$ are computed from optimal isomorphisms mapping the subcomponents of $y1$ onto the subcomponents of $x1$ and $x2$, respectively. Revising the modelling of the subcomponents of $x1$ and $y1$ may change these isomorphisms either decreasing $d_a(x1, y1)$ or increasing $d_a(x2, y1)$. This can be done by applying the following heuristics:

(i) decrease $d_a(x1, y1)$

H9: *Check if the unique subcomponents of $y1$, with respect to the similarity isomorphism between $x1$ and $y1$, are WU1, WU2, WU3 or WU4 components and deal with them if so.*

The subcomponents of $y1$, which conform to H9 can be checked in an order imposed by the following measure:

$$s(z)\big(d(z, I_s(z)\big) - d\big(z, I'_s(z)\big).$$

In this formula, z refers to a unique subcomponent of $y1$; $s(z)$ is a measure of the importance of z for $y1$, called *salience*, which is computed by the similarity analysis model; $I_s(z)$, $d(z, I_s(z))$ refer to the counterpart of z in $x1$ (i.e., nil) and the overall distance between $I_s(z)$ and z (i.e., 1), respectively; and $I'_s(z)$ and $d(z, I'_s(z))$ refer to the counterpart of z in $x2$ and the overall distance between $I'_s(z)$ and z (i.e., 1) respectively. Hence, the subcomponents of $y1$ which are subject to H9 should be considered in a sequence determined by their distance to their counterparts in $x2$, weighted by their salience for $y1$.

H10: *Check if the unique subcomponents of* $x1$ *with respect to the similarity isomorphism between* $x1$ *and* $y1$ *are WU1, WU2, WU3 or WU4 components and deal with them if so.*

The subcomponents of $x1$, which are subject to H10 can be checked in an order imposed by their salience $s(z)$, since this salience measure determines their contribution to the overall distance between $x1$ and $y1$ (function d_a in the Appendix).

H11: *Check for WC1 or WC3 subcomponents in the similarity isomorphism between* $x1$ *and* $y1$ *and deal with them if any.*

Notice that checking and possibly revising WC2 and WC4 components in $x1$ and $y1$ would not decrease $d_a(x1, y1)$, since re-mapping them would give rise to two or one unique components in the specifications, respectively.

(ii) increase $d_a(x2, y1)$

H12: *Check for WC2 or WC4 subcomponents with respect to the similarity isomorphism between* $y1$ *and* $x2$ *and deal with them if any.*

WC2 components would increase the attribution distance between $y1$ and $x2$ if left unmapped, since according to function d_a they would contribute to this distance to the maximum possible extent.

3.4. Dealing with WU3-components

H13: *Create a new component* $y1$ *with no subcomponents and deal with* $y1$ *and* $x1$ *as WU1 components.*

H13 supports the elicitation of new components in one of the specifications involved.

H14: *Check for WU1, WU2, WU3 or WU4 components in the subcomponents of* $x1$ *and deal with them if any.*

3.5. Dealing with WU4-components

Having been identified as redundant, WC4 components should be removed from their aggregating specifications. Hence

H15: *Remove* $x1$.

3.6. Dealing with WC1-components

WC1 components as counterparts of WU2 components in I_s appear for the same reasons and therefore the heuristics introduced in section 3.3 can be applied.

3.7. Dealing with WC2-components

Since the contribution of unique components to the overall distance between specifications is maximal, similarity analysis prefers to map them onto dissimilar counterparts rather than leaving them unmapped, provided that the criterion of semantic homogeneity allows it. By doing this, it minimizes the overall distance between the specifications being compared. In such circumstances it is possible that an incorrect common classification of two components might cause an undesired mapping, which can be avoided by modifying the classification of the components. This case might be subject to the heuristics H3 and H4.

3.8. Dealing with WC3-components

The WC3-components $x1$ and $y2$ of Case 6 in Fig. 2 might not have been mapped onto each other as required either because they were not identically classified or because the mapping of $x1$ onto $y1$ and $x2$ onto $y2$ had a relatively lower aggregate distance, i.e., $d(x1, y1) + d(x2, y2) < d(x1, y2) + d(x2, y1)$. In this circumstance we can apply the following heuristic:

H16: *Regard $x1$ as a WU2-component that should be mapped onto $y2$, and $y2$ as a WU2-component that should be mapped onto $x1$, and deal with them*

H16 leads to the application of the heuristics concerning the classification of components and the inequalities between their generalisation and attribution distances, discussed above.

3.9. Dealing with WC4-components

The WC4-component $x1$ of Case 7 in Fig. 2 may be dealt with as suggested by the following heuristic:

H17: *Create a new component $y2$ with no subcomponents and consider it as WU2-component.*

In the next section, we demonstrate how the previous heuristics can be used to explore ontological overlaps between specifications.

4. Scenario

We demonstrate the application of the reconciliation method using a scenario of exploring overlaps between two object-oriented specifications of library borrowers, items and their relations. These specifications overlap by including components representing library borrowers (the object types *Borrower* and *Student*), different types of library items (the object types *CopyOfBook, Publication* and their subtypes) and

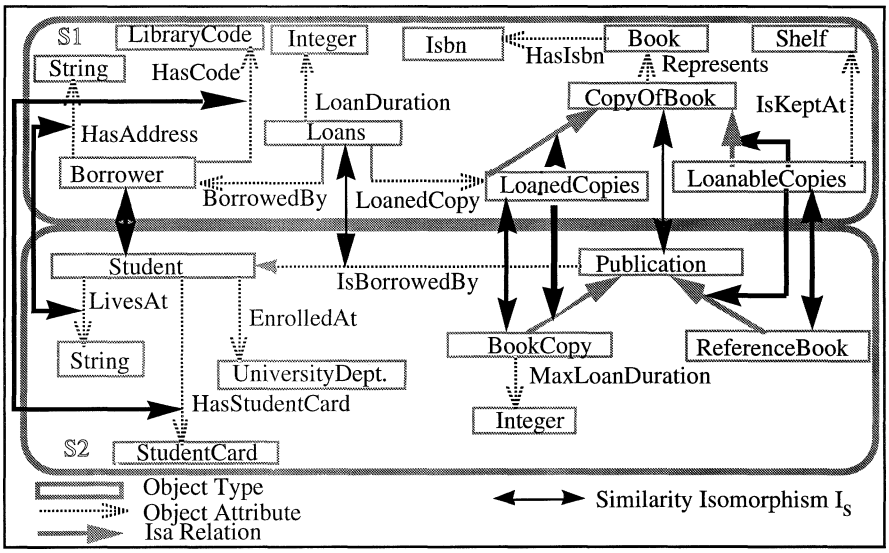

Figure 3. Scenario Step 1.

borrowing relations between them (the object type *Loans* and the object attribute *Borrows*). However, they are not modelled identically (different taxonomies of library items, different attributes for students and borrowers). Similarity analysis generates the isomorphism I_s shown in Fig. 3, driven by the identical classification and the structural similarities of the components.

In particular, I_s maps:

(1) *Borrower* onto *Student*, as the most similar pair of natural kind, agent modelling entities;

(2) *CopyOfBook* onto *Publication*, as the most similar pair of natural kind, entities;

(3) *LoanedCopies* onto *BookCopy*, as the most similar pair of natural kind, entities;

(4) *LoanableCopies* onto *ReferenceBook*, as the most similar pair of natural kind, entities;

(5) *HasCode* onto *HasStudentCard*, as the most similar pair of 1 : 1, total, onto, contemporaneous, nonhomogeneous, separable and existentially independent binary relations;

(6) *HasAddress* onto *LivesAt*, as the most similar pair of $N : M$, optional, onto, contemporaneous, nonhomogeneous, separable and existentially independent binary relations;

(7) *Loans* onto *IsBorrowedBy*, as the most similar pair of $N : 1$, optional, not onto, contemporaneous, nonhomogeneous, separable and existentially independent binary relations.

Notice that I_s has been selected among other isomorphisms that map identically classified components in S1 and S2 since it yields the minimum attribution distance (cf. function D_a in the Appendix). For instance, the attribute *LivesAt* could have been mapped onto the attribute *LoanDuration* in S1 because of their identical classifications. However, the higher similarity of the objects of *HasAddress* and *LivesAt* (these are *Student* and *Borrower*) and the identity of their values (both are strings) led analysis to their mapping as a more plausible one.

The reconciliation of S1 and S2 starts from the assessment of I_s by specification owners. Assuming that the mappings of *Borrower* onto *Student, HasCode* onto *HasStudentCard, HasAddress* onto *LivesAt* and *Loans* onto *IsBorrowedBy* are verified by specification owners as correctly reflecting ontological overlaps, we concentrate on the other associations in I_s. According to specification owners, the mappings of *CopyOfBook* onto *Publication, LoanedCopies* onto *BookCopy* and *LoanableCopies* onto *ReferenceBook* do *not* reflect correct ontological overlaps.

In particular, *CopyOfBook* should have been mapped onto *BookCopy* rather than *Publication*, as indicated by the specification owners isomorphism I_o in Fig. 4. Thus, according to our scheme *CopyOfBook* and *BookCopy* are WC3-components. The incorrect mapping might be explored by considering *CopyOfBook* as a WU2-component that should have been mapped onto *BookCopy* and vice versa, as suggested by H16. Since *CopyOfBook* and *BookCopy* had been identically classified as nominal kind entities the application of H1 and H2 does not reveal any problem in respect of the criterion of semantic homogeneity. Also, the application of H3 and H4 does not reveal any accidental incorrect common classification for *LoanedCopies* and *BookCopy* or *CopyOfBook* and *Publication* (all of them had been correctly classified as nominal kind entities). As indicated by the computed partial distances of these components,

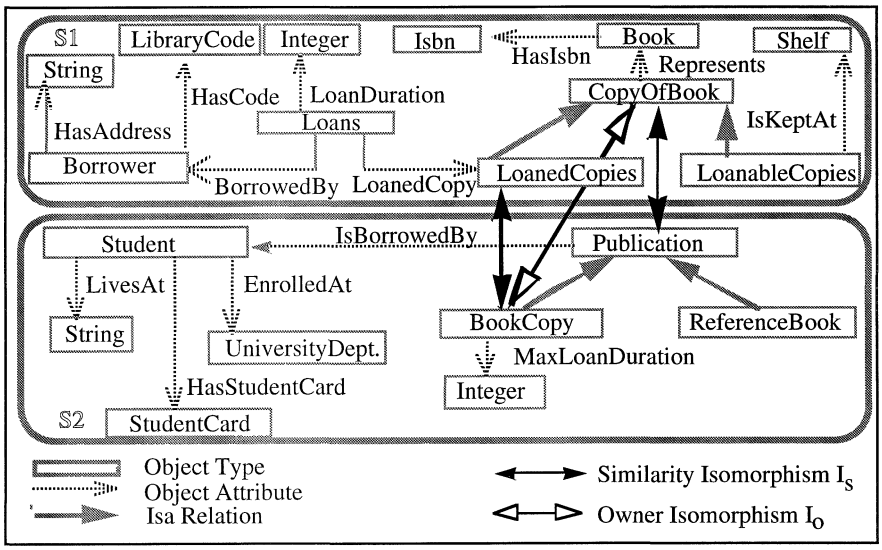

Figure 4. Scenario Step 2.

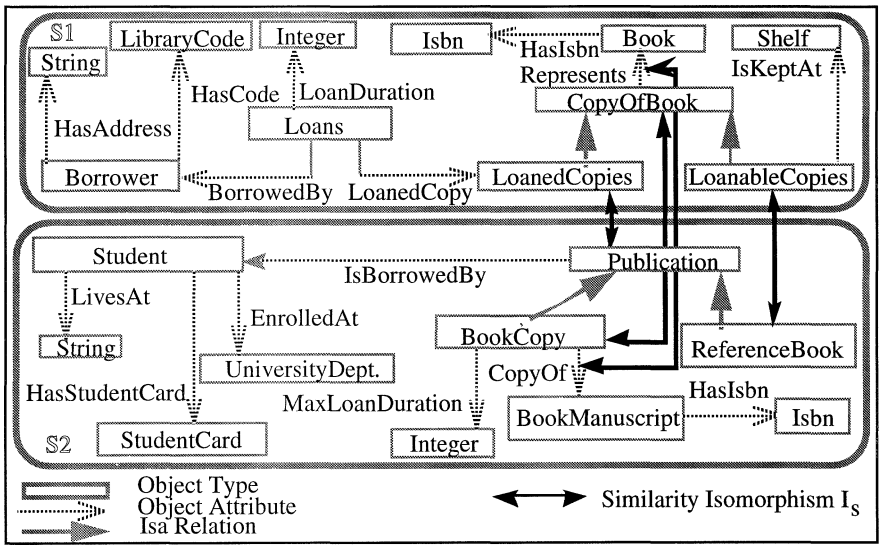

Figure 5. Scenario Step 3.

the incorrect mapping was generated because of the unequal attribution distances

$$d_a(\textit{LoanedCopies,BookCopy}) + d_a(\textit{CopyOfBook,Publication}) < d_a(\textit{CopyOfBook,BookCopy}) + d_a(\textit{LoanedCopies,Publication}).$$

To reverse this inequality we should apply H9, H10, H11 and H12.

More specifically, H10 leads to the realisation that the subcomponent (attribute) *Represents* of *CopyOfBook* is a WU3-component. In other words, it is missing from the specification of *BookCopy* in S2. By way of H13, the specification owner decides to create a new attribute, called *CopyOf*, for the object type *BookCopy*. A new object type called *BookManuscript* is also introduced as the value of this attribute. However, even after this modification similarity analysis still fails to map *Book* onto *BookManuscript* because they are not classified identically with respect to the metamodel. Treated as WU1-components, *BookManuscript* is classified identically with *Book* as a nominal kind component (due to H2). Similarly to the attribute *Represents*, the attribute *HasIsbn* of *Book* is identified as a WU3-component (given the similarity isomorphism between *Book* and *BookManuscript*) and through H13 and H2, a new attribute, having the same name and classification with it, is created for *BookManuscript*. These modifications result in the specifications and the similarity isomorphism, which is (partially) shown in Fig. 5.

However, even the new isomorphism is viewed as ontologically incorrect because of the mapping of *LoanedCopies* onto *Publication* and *LoanableCopies* onto *ReferenceBook*. As identified by the mapping of *CopyOfBook* onto *BookCopy*, *LoanedCopies* and *LoanableCopies* should correspond to subtypes of *BookCopy*. Since no

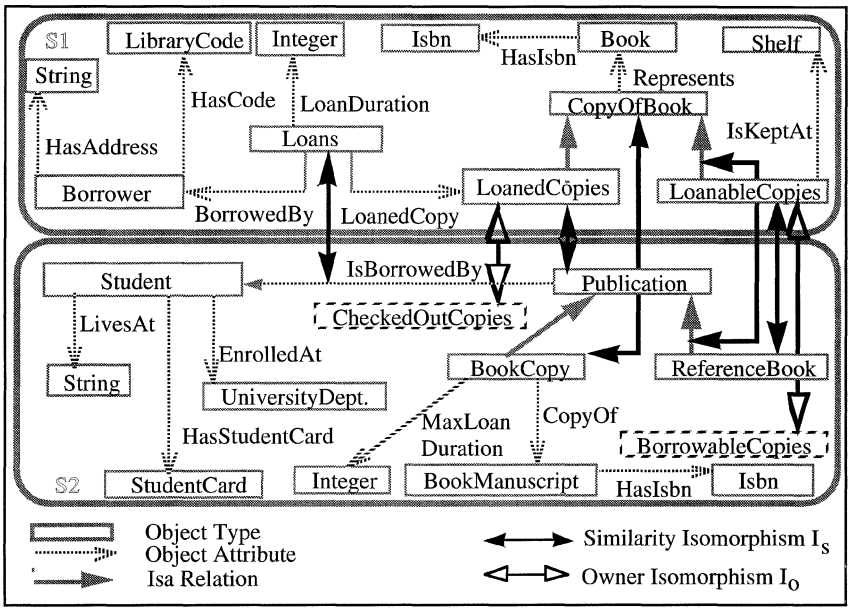

Figure 6. Scenario Step 4.

such subtypes exist in S2, *LoanedCopies* and *LoanableCopies* are considered as WC4-components. Consequently (H17), two new object types called *CheckedOutCopies* and *BorrowableCopies*, which are meant to be their counterparts are incorporated in S2 (Fig. 6). The mere incorporation of these two components is insufficient to force similarity analysis to map them as required. Thus, *CheckedOutCopies* and *BorrowableCopies* are treated as WU2-components (with respect to *LoanedCopies* and *LoanableCopies*, respectively) and consequently they are classified identically to these components (H2).

Despite this classification, similarity analysis still maps *LoanedCopies* onto *Publication* rather than *CheckedOutCopies* because of their lower attribution distance [1]

$$d_a(\textit{LoanedCopies},\textit{Publication}) = 0.502$$

and

$$d_a(\textit{LoanedCopies},\textit{CheckedOutCopies}) = 1.$$

In fact, *LoanedCopies* has one corresponding (*Loans* which corresponds to *IsBorrowedBy*) and two unique subcomponents (the *Isa* relation between *LoanedCopies* and *CopyOfBook* and the inherited attribute *Represents*) when compared to *Publication*. On the other hand, all these subcomponents are unique when compared to *CheckedOutCopies*. In trying to reverse this inequality, H10 and H13 lead to the specification

[1] In all distance computations, the normalisation parameters of the similarity model were set as follows: $b_c = b_g = b_a = b_{oo} = 0.5$ and $b_{oa} = 0.005$.

of an *Isa* relation between *CheckedOutCopies* and *BookCopy*, through which *CheckedOutCopies* inherits *IsBorrowedBy*. After these modifications, the attribution distance between *LoanedCopies* and *CheckedOutCopies* becomes lower than the attribution distance between *LoanedCopies* and *Publication*,

$$d_a(\textit{LoanedCopies,CheckedOutCopies}) = 0.34,$$

since the former pair has two corresponding subcomponents while the latter has two unique. Hence, *LoanedCopies* is mapped onto *BorrowableCopies* by similarity analysis.

Similarly, the identical classification of *BorrowableCopies* and *LoanableCopies* is insufficient to enforce their mapping since the attribution distance between *LoanableCopies* and *ReferenceBook*,

$$d_a(\textit{LoanableCopies,ReferenceBook}) = 0.602,$$

is lower than the attribution distance between *LoanableCopies* and *BorrowableCopies*, $d_a(\textit{LoanableCopies,BorrowableCopies}) = 1$. In fact, the attribution distance between *BorrowableCopies* and any other object would be equal to 1 (i.e., the maximum distance that can be computed by the similarity model) since *BorrowableCopies* has no attributes as it stands. In trying to reverse this inequality, H9 leads to the identification of the *Isa* relation connecting *LoanableCopies* with *CopyOfBook* as a WU3-component with respect to *BorrowableCopies*. By way of H13, a new *Isa* relation, connecting *BorrowableCopies* with *BookCopy* is created. This modification does not reverse the attribution distance inequality. Now, $d_a(\textit{LoanableCopies,BorrowableCopies})$ becomes equal to 0.633 as a consequence of the inheritance of the attributes *IsBorrowedBy* (from *Publication*) and *MaxLoanDuration* (from *BookCopy*) to *BorrowableCopies*. These attributes are unique as dictated by the similarity isomorphism between *BorrowableCopies* and *LoanableCopies*. As a result of H10, *IsBorrowedBy* is realized as a WU4-subcomponent (i.e., a redundant subcomponent) for *BorrowableCopies*, since a borrowing relation cannot involve items which have not been checked out from the library, as specified in S1. The operationalisation of H15 in this case involves the re-modelling of the subcomponent *IsBorrowedBy* as an attribute of *CheckedOutCopies* rather than *Publication*. Thus, it is no longer inherited by *BorrowableCopies*. Also, *MaxLoanDuration* is identified as a WU3-subcomponent (i.e., a missing subcomponent) with respect to *BorrowableCopies* and consequently (H13, H1 and H2) an analogous attribute, with the same name, is created for *CopyOfBook* and becomes an attribute of *LoanableCopies* by inheritance. These modifications reduce the attribution distance between *BorrowableCopies* and *LoanableCopies* down to 0.491 and as a result these two components are mapped onto each other by similarity analysis (see Fig. 7).

Specifications S1 and S2 have now been revised and their similarity analysis generates an isomorphism which has been verified as ontologically correct by specification owners. At this point reconciliation may stop. Along the way S1 and S2

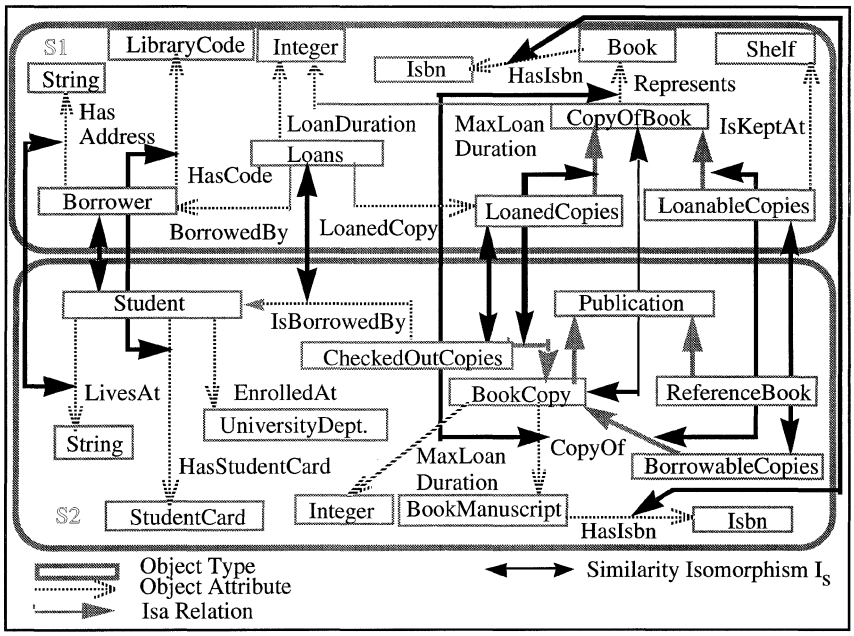

Figure 7. Scenario Step 5.

have undergone modifications that made them compatible with their owners assessment about the existence of ontological overlaps between them. New information, originally missing from them, was also elicited as a result of their comparison. Also, through the process, specification owners established a shared understanding about the ontological overlaps in their specifications and the potential for introducing further inconsistencies in them.

5. Tool support

The analysis stage in specification reconciliation is currently supported by a prototype built as a customization of the Semantic Index System (SIS), a tool for representing, storing and retrieving objects described in the Telos language [Constantopoulos and Doerr 1993]. This tool has been integrated with an implementation of the similarity model and provides queries for detecting similarities between Telos objects [Spanoudakis 1994]. The meta-model for specification analysis has been implemented as a kernel SIS object base, which is used as a schema for describing specifications. Specifications are described as SIS objects classified using this schema and are therefore amenable to similarity analysis. This process of description is supported by interactive data entry forms, built-in the SIS, and customized to support the task of classifying specification components.

To support the full reconciliation method discussed above we are building a process model, using techniques presented in [Leonhardt *et al.* 1995], describing the

activity of exploring and rectifying wrong unique and corresponding components based on the heuristics. The model will allow specification owners to switch between the stages of analysis and revision if they feel it is necessary, and guide – rather than forcing – them to adopt specific, predefined types of reconciliation [Finkelstein *et al.* 1994].

6. Related work

The detection of ontological overlap and the resolution of the inconsistencies it might cause have been concerns in requirements engineering research, due to a recent interest in requirements specification from multiple viewpoints [Finkelstein and Sommerville 1996; Nuseibeh *et al.* 1994; Maiden *et al.* 1995; Kotonya and Sommerville 1992]. They have also been central issues in obtaining and maintaining the semantic interoperability of multiple database systems [Sheth and Larson 1990; Goh *et al.* 1994].

Research in requirements engineering has mainly concentrated on the detection and resolution of logical inconsistencies, usually taking for granted the detection of ontological overlap. Some of the approaches focus on the detection of particular types of inconsistencies between specifications expressed in specific representation models [Finkelstein *et al.* 1994; Robinson and Fickas 1994; Heitmeyer *et al.* 1995; Easterbrook and Nuseibeh 1995], while other are concerned with inconsistency in general [Zave and Jackson 1993]. There has been some work on the detection of ontological overlap based either on the generation of canonical representations of specifications in a common underlying language [Johanneson 1993; Meyers and Reiss 1991] or on elaborating analogies between specifications. This elaboration has been based on matching specifications with classes of requirements engineering problems [Maiden *et al.* 1995] or on heuristics identifying analogies from the annotation of specifications with terms in domain-specific dictionaries [Leite and Freeman 1991].

The different strategies used to obtain the semantic interoperability of multiple database systems can be distinguished into the tight-coupling, loose-coupling, and knowledge based [Goh *et al.* 1994]. The tight-coupling strategies [Batini *et al.* 1986; Sheth and Larson 1990] integrate local database schemas into one or more global schemas after detecting semantic equivalencies and disparities between them. They are based on the representation of schemas in single data models and explicitly asserted relationships between local schema components (for example, inclusion dependencies, equivalence or containment relations). However, integration is not fully automated. The loose-coupling strategies detect semantic equivalencies based on annotations of local schema components with terms in shared ontologies [Bright *et al.* 1994; Sciore *et al.* 1994]. These strategies ensure the consistent exchange of semantically equivalent information by deploying conversion functions, supplied by local database systems delegates [Sciore *et al.* 1994]. Finally, the knowledge representation strategies [Collet *et al.* 1991; Arens and Knoblock 1992] transform local schemas to a global schema,

which unifies disparate interpretations and representations. This global schema is a knowledge base describing concepts in various application domains, which enables the comparison of local schemas not explicitly interrelated with each other.

7. Conclusion

The construction of complex software systems involves many agents with different perspectives or views of the system they are trying to describe, which give rise to many partial specifications (or viewpoints). Viewpoints "interfere" with each other to the extent they refer to, or assert properties of common aspects of the system under development and its domain (i.e., ontological overlap), which in turn might be inconsistent with each other. This interference needs to be "managed".

Reconciliation, the method discussed in this paper is a method of loose interference management. It detects ontological overlaps (a prerequisite for detecting inconsistencies) by analysing similarities between viewpoints and guides viewpoint owners through a process of assessing and verifying them, thus establishing a shared understanding among these owners of the potential for inconsistency. We believe that the method has promise though there is clearly considerable scope for further work. An important issue is the extension of the method so as to make it applicable to specifications of behavioural requirements. This could be achieved by extending the meta-model so as to reflect general properties of behavioural specifications. Currently, we are investigating such extensions, using as a case study the integration of various specification models of the use-case driven Object-Oriented Software Engineering method [Jacobson 1993] with our meta-model for specification analysis. We will be looking at extending the tool support and at larger scale examples which would constitute a more realistic test.

In the long-term, we envisage the computational support for our method as a component in a tool-kit developed to support the full spectrum of interference management covering specifications expressed in different languages, with different degrees of abstraction, granularity and formality, deploying different terminologies and being at different stages of development or elaboration. We believe that to cope with the diversity of the interference problem, such a tool-kit should support multiple reasoning mechanisms and/or methods; including rule-based consistency checking [Finkelstein *et al*. 1994; Easterbrook *et al*. 1994] and computer-supported human negotiation [Easterbrook 1991].

Acknowledgements

The authors would like to thank their colleagues at City University and Imperial College. In particular thanks to: Wolfgang Emmerich; Orlena Gotel; Anthony Hunter; Jeff Kramer; Bashar Nuseibeh and Steve Easterbrook. Work on this paper was funded by EPSRC (the VOILA project) and the EU (the ISI project).

Appendix. The computational model for similarity analysis

The similarity model is composed of distance measuring functions defined on Telos objects. In this Appendix, we formally introduce these objects and functions. A more detailed account of the model, including an analysis of its polynomial complexity, is given in [Spanoudakis 1994; Spanoudakis and Constantopoulos 1996].

A.1. Telos objects

Telos objects are partitioned according to their *classification level* into Tokens and Classes. Classes are further partitioned into Simple Classes, Meta Classes, Meta Meta Classes and so on. They are also partitioned according to their *role* into Individuals (objects modelling entities) and Attributes (objects modelling properties and/or relations between entities). These four basic categories of objects have the following tuple forms:

- *Individual tokens* (I_t): $o_i = [In,A]$
- *Individual classes* (I_c): $o_i = [In,Isa, A]$
- *Attribute tokens* (A_t): $o_i = [From,In,A,To]$
- *Attribute classes* (A_c): $o_i = [From,In,Isa,A,To]$

In these forms, i is an object identifier for o_i, *In* is a set of object identifiers denoting the classes of o_i (o_i is said to be an *instance* of the classes in *In*), *Isa* is a set of object identifiers denoting the superclasses of o_i, A is a set of system identifiers denoting the direct attributes (i.e., those not inherited) of o_i, *From* is the identifier of the object owning the attribute o_i and *To* is the identifier of the object being the value/range of attribute o_i.

Telos objects have *logical names* (unique to individual objects but shared by more than one attribute objects owned by distinct classes). Telos classes have *intensions* (*INT*$[i]$) including the identifiers of the attributes they introduce or inherit from their superclasses. Each Telos attribute class i has an *original class* $OC(i)$ (i.e., the most general attribute superclass of i, which has an identical logical name with it).

A.2. Distance functions

A.2.1. Identification distance

The identification distance indicates whether two objects are identical or not. Object identity depends on the equality of internal unique identifiers which are assigned to objects by the database system. Although objects with the same identifier have exactly the same value (i.e., they have the same classes, superclasses, attributes and attribute values), objects with the same values may not have the same identifier (i.e., *deep equal* but not identical objects). The identification distance distinguishes between those two cases. Formally, this distance is defined as follows:

Definition 1. The identification distance d_{id} between two objects o_i and o_j is defined by:

$$d_{\text{id}}(o_i, o_j) = \begin{cases} 0 & \text{if } i = j, \\ 1 & \text{if } i \neq j. \end{cases}$$

A.2.2. Classification distance

The classification distance between two objects is measured by identifying and measuring the importance of their non-common classes. Thus, it gives an account of how two specifications differ with respect to the properties expressed by the classes of the meta-model. The importance of each class is measured by its specialisation depth, *SD*(x), which is formally defined as follows:

Definition 2. *SD*(x) is the maximum length (number of links) of the paths connecting class x with the most general class of its generalisation taxonomy, called *specialisation depth* of x.

Thus, non-common classes, which are placed at higher levels in generalisation taxonomies are considered as more important than those placed at lower levels. Given *SD*(x), the classification distance is defined as follows:

Definition 3. The classification distance d_c between two objects o_i and o_j is defined by:

$$d_c(o_i, o_j) = \big(b_c D_c(o_i, o_j)\big) / \big(b_c D_c(o_i, o_j) + 1\big), \quad b_c \in \mathbb{R}^+,$$
$$D_c(o_i, o_j) = \sum_{x \in NCC_{ij}} SD(x)^{-1}, \quad NCC_{ij} = (o_i.In - o_j.In) \cup (o_j.In - o_i.In),$$

b_c is a normalization parameter evaluated so that d_c equals 0.5 when D_c takes its average value given a specific set of objects (i.e., a context-sensitive estimation of the classification distance).

A.2.3. Generalisation distance

The generalisation distance provides an account of the semantic differences of two individual objects as evidenced from their non-common superclasses. Each of these superclasses is weighted by its specialisation depth, like the non-common classes in the classification distance. The generalisation distance between attribute classes depends on the identity of their original classes and distinguishes between refined specialisations of the same attribute and specialisations between attributes with shared but non-identical semantics.

Definition 4. The generalisation distance d_g between two objects o_i and o_j is defined by:

$$d_g(o_i, o_j) = \begin{cases} \big(b_g D_g(o_i, o_j)\big)/\big(b_g D_g(o_i, o_j) + 1\big),\ b_g \in \mathbb{R}^+ & \text{if } o_i, o_j \in I_c, \\ d_o(o_i, o_j) & \text{if } o_i, o_j \in A_c, \end{cases}$$

$$D_g(o_i, o_j) = \sum_{x \in NCS_{ij}} \mathit{SD}(x)^{-1},$$

$$\mathit{NCS}_{ij} = (o_i.\mathit{Isa} - o_j.\mathit{Isa}) \cup (o_j.\mathit{Isa} - o_i.\mathit{Isa}) \cup \{i, j\},$$

$$d_o(o_i, o_j) = \begin{cases} 0 & \text{if } \mathit{OC}(i) = \mathit{OC}(j), \\ 1 & \text{if } \mathit{OC}(i) \neq \mathit{OC}(j), \end{cases}$$

b_g is similar to and evaluated like b_c in Definition 3.

A.2.4. Attribution distance

The attribution distance is estimated by searching for a minimum distance isomorphism between the attributes of two objects. Mappings are only considered between attributes, which are instances of the same original attribute classes. Thus, only specification components which belong to exactly the same classes of the meta-model, and therefore share exactly the same semantic properties, can be mapped onto each other. The attribution distance takes into account the overall distance between the object-values of attributes (Definitions 5 and 6 below). Thus, it generates minimum distance isomorphisms among the attributes of these value-objects and recursively among the attributes at all the successive levels of their decomposition-closures. In doing so, it produces a detailed account of the structural resemblances between two specifications. Formally, the attribution distance is defined as follows:

Definition 5. The attribution distance d_a between two objects o_i and o_j is defined by:

$$d_a(o_i, o_j) = \big(b_a D_a(o_i, o_j)\big)/\big(b_a D_a(o_i, o_j) + 1\big), \quad b_a \in \mathbb{R}^+,$$

$$D_a(o_i, o_j) = \begin{cases} \infty & \text{if } o_i.A = \varnothing \text{ or } o_j.A = \varnothing, \\ \min\limits_{m \in I(o_i, o_j)} \Big(\sum\limits_{(x_1, x_2) \in m} s(x_1) s(x_2) d(x_1, x_2) \\ \quad + \sum\limits_{x_3 \in o_i[m]} s(x_3)^2 + \sum\limits_{x_4 \in o_j[m]} s(x_4)^2 \Big), & \text{otherwise,} \end{cases}$$

where
$I(o_i, o_j)$ is the set of all the possible morphisms between the semantically homogeneous attributes of o_i and o_j (two attribute objects k and l are semantically homogeneous if and only if $\mathit{OCL}[k] = \mathit{OCL}[l]$ where $\mathit{OCL}[x] = \{y \mid (y = \mathit{OC}(z)$ and $z \in o_x.\mathit{In}\}$),

$o_i[m]$ $(o_j[m])$ is the set with the attributes of o_i (o_j) that map onto no attribute of $o_j(o_i)$ given the isomorphism m,
$s(x)$ is the salience of attribute class x computed as described in [Spanoudakis and Constantopoulos 1996].

b_a is similar to and evaluated as b_c in Definition 3.

A.2.5. Overall distance

The overall object distance aggregates the partial identification, classification and generalisation distances between two objects. Thus, it gives an overall account of both the semantic and the structural differences between two objects, which is particularly useful in cases where they are incompletely described with respect to classification and generalisation relations or attributes. Formally, the overall distance is defined as follows:

Definition 6. The overall distance d between two objects o_i and o_j is defined by:

$$d(o_i, o_j) = \big(b_{oo} D(o_i, o_j)\big) / \big(b_{oo} D(o_i, o_j) + 1\big), \quad b_{oo} \in \mathbb{R}^+, \quad \text{where}$$
$$D(o_i, o_j) = \big(d_{\text{id}}(o_i, o_j)^2 + d_c(o_i, o_j)^2 + d_g(o_i, o_j)^2 + d_a(o_i, o_j)^2$$
$$+ d_c(o_i, o_j) d_g(o_i, o_j) + d_c(o_i, o_j) d_a(o_i, o_j) + d_g(o_i, o_j) d_a(o_i, o_j)\big)^{1/2}$$
$$\text{if } o_i, o_j \in (I_t \cup I_c), \text{ and}$$

$$d(o_i, o_j) = \big(b_{oa} D(o_i, o_j)\big) / \big(b_{oa} D(o_i, o_j) + 1\big), \quad b_{oa} \in \mathbb{R}^+, \quad \text{where}$$
$$D(o_i, o_j) = \big(d_{\text{id}}(o_i, o_j)^2 + d_c(o_i, o_j)^2 + 36 d_g(o_i, o_j)^2 + d_a(o_i, o_j)^2 + d(o_i.\mathit{To}, o_j.\mathit{To})^2$$
$$+ 12 d_c(o_i, o_j) d_g(o_i, o_j) + d_c(o_i, o_j) d_a(o_i, o_j) + 12 d_g(o_i, o_j) d_a(o_i, o_j)\big)^{1/2}$$
$$\text{if } o_i, o_j \in (A_t \cup A_c).$$

D has a quadric functional form because of experimental evidence about statistically significant correlation between d_c, d_g and d_a [Spanoudakis and Constantopoulos 1996]. The relatively higher coefficients of products having d_g as a factor (i.e., 36, 12) ensure that attributes with the same original class (whose generalisation distance is equal to 0 according to Definition 4) will necessarily be mapped to each other, when comparing the objects to which they apply. b_{oo} and b_{oa} are similar to and evaluated as b_c in Definition 3.

References*

Arens, Y. and C. Knoblock (1992), "Planning and Reformulating Queries for Semantically-Modeled Multidatabase Systems," In *Proceedings of the 1st International Conference on Information and Knowledge Management*, pp. 92–101.

Batini, C., M. Lenzerini, and S. Navathe (1986), "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys 18*, 4, 322–364.

Bright, M., A. Hurson, and P. Pakzad (1994), "Automated Resolution of Semantic Heterogeneity in Multidatabases," *ACM Transactions on Database Systems 19*, 2, 212–253.

Collet, C., N. Huhns, and W.-M. Shen (1991), "Resource Integration Using a Large Knowledge Base in Carnot," *IEEE Transactions on Computers 24*, 12, 55–63.

Constantopoulos, P. and M. Doerr (1993), "The Semantic Index System: A Brief Presentation," Institute of Computer Science, Foundation for Research and Technology-Hellas, Heraklion, Crete, Greece. (Available by ftp from: *http://www.ics.forth.gr/proj/isst/Systems/SIS/index.html*).

Easterbrook, S. (1991), "Handling Conflict between Domain Descriptions with Computer-Supported Negotiation," *Knowledge Acquisition 3*, 255–289.

Easterbrook, S., A. Finkelstein, J. Kramer, and B. Nuseibeh (1994), "Co-Ordinating Distributed View-Points: The Anatomy of a Consistency Check," *International Journal on Concurrent Engineering: Research and Applications 2*, 3, CERA Institute, USA, 209–222.

Easterbrook, S. and B. Nuseibeh (1995), "Managing Inconsistencies in an Evolving Specification," In *Proceedings of the IEEE International Conference on Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, CA, pp. 48–55.

Finkelstein, A., D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh (1994), "Inconsistency Handling in Multi-Perspective Specifications," *IEEE Transactions on Software Engineering 20*, 8, 569–578.

Finkelstein, A. and I. Sommerville (1996), "The Viewpoints FAQ," *Software Engineering Journal: Special Issue on Viewpoints for Software Engineering 11*, 1, 2–4.

Goh, C., S. Madnick, and M. Siegel (1994), "Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment," In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, Gaithersurg, MR, USA, pp. 337–346.

Heitmeyer, C., B. Law, and D. Kiskis (1995), "Consistency Checking of SCR-Style Requirements Specifications," In *Proceedings of the IEEE International Conference on Requirements Engineering*, York, England, pp. 56–63.

Hunter, A. and B. Nuseibeh (1995), "Managing Inconsistent Specifications: Reasoning, Analysis and Action," Technical Report TR-95/15, Department of Computing, Imperial College, London, UK.

Jacobson, I. (1993), *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, New York, NY.

Johanneson, P. (1993), "Schema Transformations as an Aid in View Integration," In *Proceedings of CAiSE'93*, Lecture Notes in Comput. Sci. 685, Springer, Berlin, Germany, pp. 144–151.

Kotonya, G. and I. Sommerville (1992), "Viewpoints for Requirements Definition," *Software Engineering Journal 7*, 6, 375–387.

Leite, J. and P. Freeman (1991), "Requirements Validation Through Viewpoint Resolution," *IEEE Transactions on Software Engineering 17*, 12, 1253–1269.

Leonhard, U., A. Finkelstein, J. Kramer, and B. Nuseibeh. (1995), "Decentralised Process Enactment in a Multi-Perspective Development Environment," In *Proceedings of the 17th International Conference on Software Engineering* (*ICSE-17*), IEEE Computer Society Press, Los Alamitos, CA, pp. 255–264.

* Note: Copies of many papers by the authors cited below are available from their WWW pages at http://www.cs.city.ac.uk/finger/~acwf or http://www.cs.city.ac.uk/homes/gespan/info.html.

Maiden, N., P. Assenova, P. Constantopoulos, M. Jarke, P. Johanneson, H. Nissen, G. Spanoudakis, and A. Sutcliffe (1995), "Computational Mechanisms for Distributed Requirements Engineering," In *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering* (*SEKE'95*), Knowledge Systems Institute, New York, NY, pp. 8–16.

Meyers, S. and S. Reiss (1991), "A System for Multiparadigm Development of Software Systems," In *Proceedings of the 6th International Workshop on Software Specification and Design* (*IWSSD-6*), IEEE Computer Society Press, Los Alamitos, CA, pp. 202–209.

Motschnig-Pitrik, P. (1993), "The Semantics of Parts vs. Aggregates in Data Knowledge Modeling," In *Proceedings of CAiSE'93*, Lecture Notes in Comput. Sci. 685, Springer, Berlin, Germany, pp. 352–373.

Mylopoulos, J., A. Borgida, M. Jarke, and M. Koubarakis (1990), "Telos: Representing Knowledge About Information Systems," *ACM Transactions on Information Systems 8*, 4, 325–362.

Nuseibeh, B., J. Kramer, and A. Finkelstein (1994), "A Framework for Expressing the Relationship Between Multiple Views in Requirements Specification," *IEEE Transactions on Software Engineering 20*, 10, 760–773.

Robinson, W. and S. Fickas (1994), "Supporting Multi-Perspective Requirements Engineering," In *Proceedings of the IEEE Conference on Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, CA, pp. 206–215.

Sciore, E., E. Siegel, and A. Rosenthal (1994), "Using Semantic Values to Facilitate the Interoperability Among Heterogeneous Information Systems," *ACM Transactions on Database Systems 19*, 2, 254–290.

Sheth, A. and J. Larson (1990), "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases," *ACM Computing Surveys 22*, 3, 183–236.

Spanoudakis, G. (1994), "Analogical Similarity of Objects: A Conceptual Modelling Approach," PhD Dissertation, Department of Computer Science, University of Crete, Heraklion, Greece (available by ftp from: *http://web.cs.city.ac.uk/homes/gespan/info.html*).

Spanoudakis, G. and P. Constantopoulos (1994a), "Similarity for Analogical Software Reuse: A Computational Model," In *Proceedings of the 11th European Conference on Artificial Intelligence* (*ECAI'94*), John Wiley and Sons, New York, NY, pp. 18–22.

Spanoudakis, G. and P. Constantopoulos (1994b), "On Evidential Feature Salience," In *Proceedings of the 5th International Conference on Database and Expert Systems Applications* (*DEXA'94*), Springer, Berlin, Germany, pp. 153–162.

Spanoudakis, G. and P. Constantopoulos (1995), "Integrating Specifications: A Similarity Reasoning Approach," *Automated Software Engineering Journal 2*, 4, 311–342.

Spanoudakis, G. and P. Constantopoulos (1996), "Elaborating Analogies from Conceptual Models," *International Journal of Intelligent Systems 11*, 11, 917–974.

Storey, V. (1993), "Understanding Semantic Relations," *Journal of Very Large Data Bases 3*, 455–488.

Zave, P. and M. Jackson (1993), "Conjunction as Composition," *ACM Transactions on Software Engineering and Methodology 2*, 4, 379–411.