Similarity for Analogical Software Reuse: A Conceptual Modelling Approach

George Spanoudakis and Panos Constantopoulos

Department of Computer Science, University of Crete and
Institute of Computer Science
Foundation of Research and Technology - Hellas
Heraklion, Crete, Greece

Abstract. We present our approach to defining similarity between software artifacts and discuss its potential exploitation in software reuse by analogy. We first establish properties of similarity which support its role in retrieving and mapping software descriptions. Then we develop a systematic basis for comparison within a fairly general conceptual modelling framework, whereby comparable elements of the descriptions of software objects and corresponding similarity criteria are identified. Finally, a general form of distance metrics for the computation of similarity measures is defined.

1. Introduction: The Analogical Based Approach to Software Reuse

The reuse based approach to software development has been realized as a form of analogical problem solving, where a reasoner, the software engineer, familiar with a variety of source cases(i.e development histories and systems) attempts to transfer properties, relations and solutions from them to a target system under development[24,26].

This realization brings reuse in a ground rich in theoretical predictions and experimental evidence from a variety of scientific disciplines(AI, Psychology and Cognitive Science), offering a generic process schema of analogical reasoning, with 4 stages:

1. the *retrieval* of a set of source cases given a target description ;

2. the *mapping* of knowledge from some of the retrieved sources to the target on the basis of analogies between them, established at the same stage ;

3. the *evaluation* of the transferred knowledge in terms of consistency and pragmatic utility ; and

4. the *consolidation* of the final outcome into structures of knowledge, supporting similar forms of reasoning in the future.

Research into reuse can benefit from this conceptual framework. In fact, the solutions to the operational problems of reusability(i.e retrieval, comprehensibility, modifiability and composition[4]) offered so far, can be reviewed, enhanced with new ones and ideally integrated into a generic and theoretically plausible methodology according to the predictions of analogical reasoning.

Adopting this approach, we focus on the problem of defining a quantitative similarity relation between descriptions of software artifacts so as to promote their analogical reuse.

The potential of similarity has been pointed out in both the literature relevant to reuse[5,7,8,9] and the literature relevant to analogical reasoning[1,16,30]. Also, heuristic realizations of the concept have been exploited in validating and integrating specifications of requirements(e.g viewpoint resolution[22]).

Similarity has a clear role in analogical reasoning. As a fine evaluator of source cases initially selected by some coarser search-method(e.g constrained spreading of activation[6]) or as a basic retrieval mechanism, it can provide effective trial hints to the stage of mapping that follows retrieval. However, the effectiveness of similarity depends on its very definition and whether it estimates correctly the importance of the case features.

In this paper, we present our on-going work on the concept of similarity. The final aim is the formulation of a computational model of similarity, in a way consistent with properties inherent to relevant kinds of human processing[33,36] and dealing effectively with analogical reuse.

2. A General Review of Similarity

Similarity is described as a relation determined by some flexible comparison between the distinct constituents of two entities (i.e situations, cases, natural or nominal-kind objects)[25,31,33,36,38,40]. From a quantitative viewpoint, the result of this comparison may be interpreted in two ways:

i. as a measure of closeness in some abstract space[25,36] and

ii. as a probability that the objects under comparison, would resemble each other even if their possibly missing constituents were considered as well[10,30].

Both the representation of objects and the nature of the comparisons are approached differently in the literature.

We can dinstinguish between two different representations. The first assumes a predefined set of features adequate for describing any object within a given domain of discourse. while the second assumes that objects are representable through their classification into a set of mutually disjoint and exhaustive classes.

Object comparisons, may also take two forms:

(1) the *exact-matching form*, which results into three sets of object-features:

$$i.\ the\ common\ features:\ S1 = F(o1) \cap F(o2)$$

$$ii. the\ distinctive\ features\ of\ the\ first\ object: S2 = F(o1) - F(o2)\ and,$$

$$iii. the\ distinctive\ features\ \ the\ second\ object: S3 = F(o2) - F(o1)$$

Given these three sets, the evaluation of similarity, in general has one of the following functional forms:

i. F(S1, S2, S3) that takes into account both the common and the distinct features of the objects. Typical instances of this form are[36]:

$$a.\text{the ratio model}: \frac{wf(S1)}{af(S2)+bf(S3)} \text{ and},$$

$$b.\text{the contrast model}: wf(S1)-af(S2)-bf(S3)$$

ii. F(S1) that takes into account only the common features of the two objects. In this category, we can classify the frequency model of similarity[33]:

$$f(\ |C|\ ,\ |U|\ )$$

where

|C| is the cardinality of the common class of two objects

|U| is the cardinality of the entire universe of objects and,

f is a function increasing in |C| and decreasing in |U|

(2) the *distance-based form*, which measures the distance of features according to the types of their domains(e.g nominal, totally ordered domains). Typical such metrics include:

i. the identity function for nominal domains and

ii. the absolute difference function for totally ordered domains[2,10]:

d(v1,v2) = |order(v1) - order(v2)|

3. TELOS: A Representational Framework for Similarity Evaluation

In this section we briefly review the structural part of the TELOS data model[27,37] which will be the basis for representation in our framework. TELOS has been chosen because it subsumes the structural constructs of other object oriented data models, and has been acknowledged as language for describing software artifacts[7].

It provides three basic abstractions, namely classification, generalization and attribution.

Classification defines an infinite dimension along which objects can be classified into built-in disjoint classes that distinguish between the successive levels of classification(i.e Token class, S_Class class, M1_Class class, M2_Class class and so on). In addition objects can be classified as instances of other user-defined classes.

Classes in TELOS can be generalized into other classes, through Isa-relations. These relations have a set inclusion semantics, are transitive and hold only between classes of the same classification level. Moreover, they allow a strict and multiple inheritance of attributes from the superclasses to the subclasses.

The attribution mechanism allows the attachment of attributes to objects. Attributes in TELOS are also objects. Thus they can be classified into attribute classes, generalized, and have attributes of their own. The term attribute captures both classes and tokens of attributes. Attribute classes may be single-valued or multi-valued.

Every TELOS object is associated with two unique identifiers. The first of them is known as *system identifier*(i.e a surrogate generated automatically by the system). The second identifier reffered to as *logical name* accomplishes logical references to objects. Although it may be changed during the life-cycle of an object, a logical name ought to be system-wide unique. The logical names of attribute objects are composed of the logical names of their possessing objects and logical referents attached directly to them, called *labels*.

According to the previous overview, four basic forms of TELOS objects can be distinguished:

1. Entity Tokens: ETi = [Id(i), In(i), A(i)]

2. Attribute Tokens: ATi = [Id(i), Id(From(i)), In(i), A(i), Id(To(i))]

3. Entity Classes: ECj = [Id(j), In(j), Isa(j), A(j)]

4. Attribute Classes: ACj = [Id(j), Id(From(j)), In(j), Isa(j), A(j), Id(To(j))]

where

i. Id(x) is a system identifier denoting the object Ox

ii. In(x) is a set of system identifiers denoting the classes of the object Ox

iii. Isa(x) is a set of system identifiers denoting the direct superclasses of the class Ox

iv. A(x) is a set of system identifiers denoting the attributes attached to the object Ox

v. From(x) denotes the object possessing the attribute Ox and,

vi. To(x) denotes the object pointed to by the attribute Ox

Two more issues regarding the topology of the Isa-graphs and the inherent assumptions of the inheritance in TELOS matter from a similarity perspective.

In TELOS, an Isa-graph generally consists of M disjoint subgraphs, where M is the number of classification levels used in the particular schema(see figure 1). These graphs are disjoint because the Isa relations are restricted only between classes of the same classification level.

The set-inclusion semantics of the Isa-relation make it a partial-order relation[20]. Moreover, at each level of classification TELOS provides a most general class, whose extension contains all the objects of the lower level (see Token, S_Class and M1_Class in figure 1). Consequently due to the set-inclusion semantics of the Isa-relations, these classes are regarded as superclasses of any other class at the same classification level. Thus two classes in an Isa subgraph will always have a common superclass.

The key hypothesis of the TELOS inheritance mechanism is that the identity of attribute-labels along an Isa-path implies the semantic identity of the relevant attribute-objects. This, due to the strict inheritance, has the consequence that attributes in subclasses, having the same labels with attributes in their superclasses, can only specialize them(see

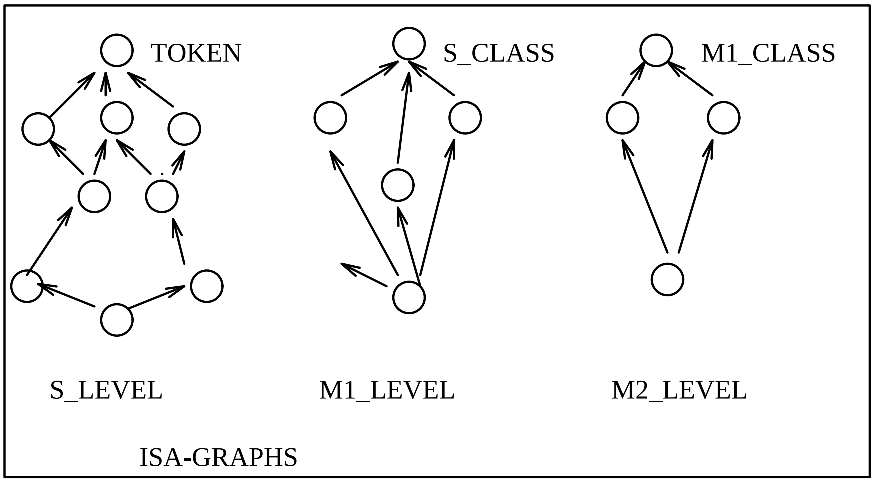

Fig. 1. Distinct TELOS Isa-graphs

figure 3).

4. Principles of Similarity Computation

In this section we formulate a set of principles underlying the computation of similarity, justified by the ways in which humans perceive similarities and by requirements of analogical reasoning.

4.1. The Principle of Ontological Uniformity

Objects may correspond to different levels of abstraction expressed through their classification levels in TELOS. Roughly speaking, objects may refer to atomic real world entities(i.e Tokens), to abstracted classes of atomic entities(i.e S_Classes), to models for such abstractions(i.e M1_Classes) and so on. In this object space, it would be senseless to compare entities of different ontologies. Any such comparison would be ad hoc and therefore not semantically interpretable. Hence, inter-object comparisons are restricted according to the following principle of *Ontological Uniformity*:

(P1) Similarity comparisons are only valid between objects of the same classification level

Ontological uniformity seems to be arguable in concept formation[12] where the term similarity refers to comparisons between objects of successive levels of classification. However, these comparisons have a different objective. In having to classify an object as an instance of another they end up in a true/false result combining the membership of the observed attribute values of the lower object to the predicted domains of attributes in the higher one.

Notice that according to relevant studies[34,41], when humans generate concepts, the predicted comparisons do not involve the objects of the higher level directly. In some cases, are carried out implicitly, through one or more prototypes(i.e objects having the greater resemblances with the rest of the instances of the concept/class) of the relevant concept(figure 2, case a)[34,41]. In other cases, the comparison is performed in an aggregate fashion, taking into account the resemblances of the object to be classified with all the other known instances of the concept(figure 2, case b)[29,36]. Finally, there exist cases where the comparison is carried out through an exemplar-based representation of the concept(i.e a representation consisting of the most prototypical instance or instances of the concept with no abstracted properties, figure 2, case c).

These findings about the comparisons involved in concept formation by humans favour the principle of ontological uniformity.

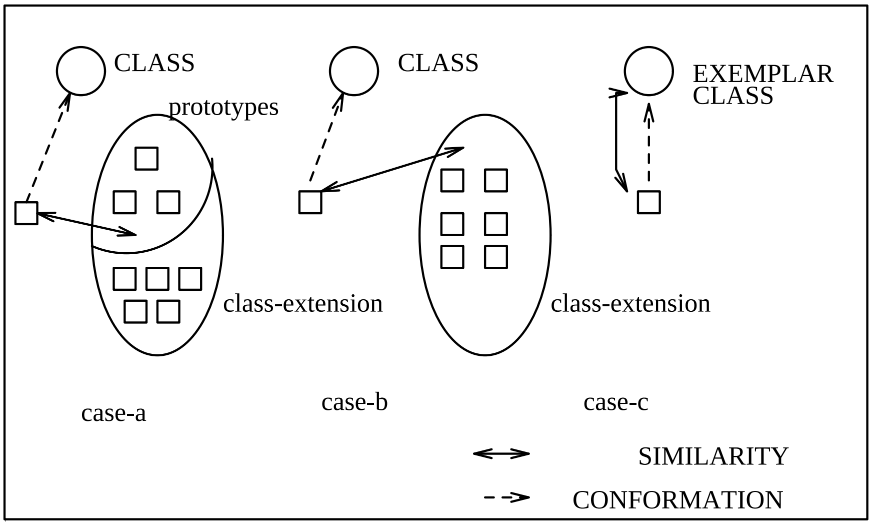

Fig. 2. Comparisons between Classes and Instances

4.2. The Principle of Partially Uniform Representation

A critical issue for the evaluation of similarity is the uniformity of the object representations(i.e whether objects are representable through a fixed set of features or not). Such a hypothesis is reasonable in narrow, well-defined and rather mature domains but questionable in the domain of software reuse. In this area, exist objects from different domains(e.g diverse application areas of software), described via non-standard and possibly not well-understood properties, predicted by models reflecting diverse viewpoints[7,11,27,37]. Such objects are only partially uniform. An example of partial uniformity is presented in figure 3.

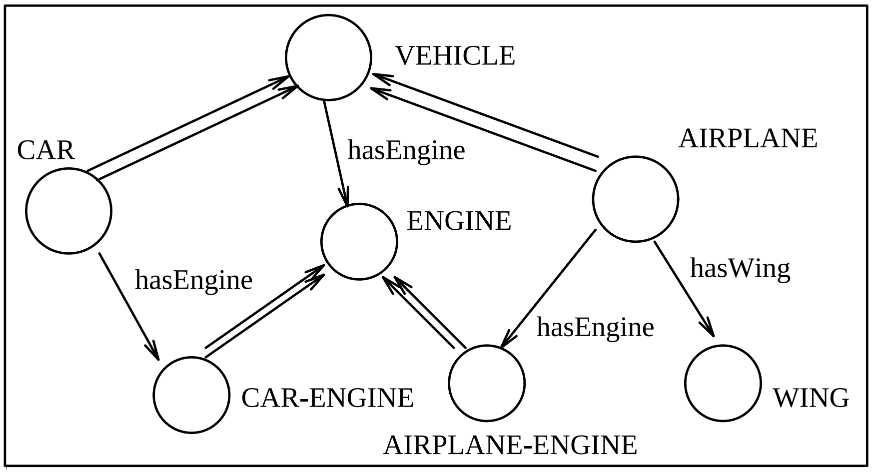

Fig. 3. Isa-graphs and Inheritance

The two different vehicle types, the airplane and the car are uniform in sharing the possession of an engine(despite the different engine types) and diverse with respect to the wings.

Partial uniformity restricts object comparisons only between their common attributes under a global perspective of analogical reasoning. During the early stage of retrieval, two basic types of errors may occur. The first is the rejection of sources that could later enable sound and useful analogical transfers(*errors of type A*). The second captures cases where retrieved candidate sources, do not enable any acceptable transfer(*errors of type B*). Errors of type B increase the overall computational cost of the process, since they enlarge the set of the cases to be considered for transfer during the mapping stage, but they are recoverable during justification. On the contrary it would be impossible to recover from errors of type A. Therefore these errors are more important than errors of type B.

Consequently, comparisons between the uncommon attributes of two objects should not be attempted. Such comparisons necessitate ad hoc matchings between ordinary and artificial values, that can only increase the dissimilarity of the involved objects, while not contributing in a positive way to any transfer.

The previous analysis is summarized into the principle of *Partially Uniform Representation*, stating that:

(P2) Objects have only partially uniform representations. The non-uniform parts of their representation must be excluded from similarity comparison

4.3. The Normalization Principle

Since distance measures subsume exact matching(see the fourth axiom below) they must be preferred as a basis for estimating similarity. We can define an overall distance metric D as an aggregate function over partial metrics, devised in accordance with the semantics of the three distinct abstractions composing the object descriptions.

D is a mapping of the form:

D: O x O ----> R

where O is the set of all objects in the context of similarity estimation, R is the set of Reals and D obeys the known metric axioms[21]:

(1) D(x,y) >= D(x,x) = 0

(2) D(x,y) = D(y,x)

(3) D(x.z) <= D(x,y) + D(y,z) and.

(4) D(x,y) = 0 ==> x = y

Similarity has been viewed as a monotonically decreasing function f of distance D [30] that may be defined over absolute or relative distance measures. Certain characteristics of the abstractions over which the absolute distances must eventually be defined, strongly favor relative measures. Generally speaking, the elementary absolute distance metrics must be defined over Isa-graphs or domains of attributes that may be nominal, partially ordered or totally ordered(i.e linear). In all but the case of the nominal domains, which suggests an identity distance function, the relevant domains are not expected to have equal widths[30] and the Isa-graphs are expected to have diverse coarsenesses. Therefore, normalizations of absolute metrics are necessary for filtering out differences arising due to coarser and finer representations.

This requirement is summarized in the following principle of *Normalization*:

(P3) Similarity(S) is a monotonically decreasing function of a normalized Distance measure(Dr)

4.4. The Importance of Features in Similarity Evaluation

The distinct features of objects have a different impact(known as salience) in both the evaluation of similarity and the success of analogical transfer. This impact varies according to the domain dominance, the pragmatic utility and the classificatory significance of a feature. Since its overestimation or underestimation may prevent analogical transfer or enable erroneous analogies[19,28,32], salience must be carefully quantified in similarity estimates.

The domain dominance of features

The domain dominance reflects the causality of a feature within some domain. Causality is determined by the dependence of values or even the presence of other features on a particular feature and has a direct effect on the mapping stage of analogical reasoning(dominance contributes to transfer of entire constellations of knowledge pieces).

There exist two different viewpoints to the evaluation of the domain dominance. The first one[16,30,39] realizes it as presence in the antecedent parts of implications or in integrity constraints expressing inter-feature dependencies for a particular domain. The second approach relies on syntactic aspects of object representations. It distinguishes between dominant relations and other not dominant representational elements(i.e properties)[13,14].

Certain observations about the encoding of object knowledge bases make plausible the evaluation of dominance on the basis of syntactic elements. Studies of the surface/structure paradox of analogical reasoning[15], indicate that the encoding of some domain around dominant features instead of superficial ones is subject to the training and the expertise of the encoder in it. In our case both the training and the expertise conditions of encoding are satisfied. Software repositories are normally developed by experts and undergo constant improvements(e.g reverse engineering) during their life-cycle(see [8] for a similar viewpoint). Consequently domain dominant features will be prevalent in them.

Hence the principle of *the Domain Dominant Schema*:

(P4) The domain dominance of a feature can be determined from the schema of the relevant object knowledge base.

The pragmatic utility of features

The causal relations of features to goal attainment in analogical reasoning designate their pragmatic utility. Features may predict the applicability of some method for achieving a goal, explain the success or failure of a certain solution, describe unusual outcomes of solutions or be totally irrelevant to goals and solutions. In all but the forth of these cases, they have a high pragmatic utility.

However, it is really difficult to distinguish between pragmatic utility and domain dominance in the absence of explicit information about the former(e.g Goal Dependency Graphs in [35]). In practice, domain dominance correlates with pragmatic utility since general causality is likely to imply to goal relevancy. Therefore, without any important loss of information, we can rely on domain dominance in estimating pragmatic utility.

Thus, the principle of *the Pragmatic Utility Subsumption* suggests:

(P5) The pragmatic utility of a feature can be approximated by its domain dominance

Proposed conceptual schemas for software repositories, which capture pragmatically important features(e.g correspondence links in the SIB[7], abstract domain classes in [24]) justify this principle.

The classificatory significance of features

The ability of features to produce classification schemas, optimal with respect to certain criteria(e.g predictability in concept formation, precision in information retrieval) constitutes their classificatory significance.

In this framework, the sensitivity to errors of type A suggests a reasonable optimality criterion for a classification schema: the minimization of the probability of errors of type A. Consequently we can define the classificatory significance of a feature through the following principle of *Classification Optimality*:

(P6) The classificatory significance of a feature depends on its ability to produce classification schemas minimizing the probability of errors of type A in the analogical reasoning process.

5. Establishing A Basis for Comparison

The classification, the generalization, the attribution and the distinct identifiers are really expressive in describing objects but impose the problem of deciding how to compare them.

The different nature of the various object elements suggests comparisons only between elements of the same type if we want to have a clear semantic basis for defining distance metrics. Therefore, we adopt the following top-level pairs of comparison, on the basis of our representational model:

1. Comparisons between Classifications:

    In(x) <---> In(y) {for any pair of objects (x,y)}

2. Comparisons between Generalizations:

    Isa(x) <---> Isa(y) {for any pair of classes (c1,c2)}

3. Comparisons between Attributes:

    A(x) <---> A(y) {for any pair of objects (x,y)}

4. Comparisons between System Identifiers:

    Id(x) <---> Id(y) {for any pair of objects (x,y)}

    Id(From(x)) <---> Id(From(y)) {for any pair of attributes (x,y)}

    Id(To(x)) <---> Id(To(y)) {for any pair of attributes (x,y)}

These top-level comparisons must be further elaborated in the case of the attributes, which can be utilized in representing properties and/or relations with different semantics (e.g Car.partOf, Car.mileage). A distinction is made between attribute classes and attribute tokens, due to their different inheritance properties and the roles of their labels.

5.1. Comparisons Between Attribute Classes

Recall that attribute classes with the same labels along an Isa-path are perceived as being semantically identical and consequently only specializations of their specifications are permitted. Note also, that in the case of unordered classes(with respect to Isa relations) label equality has no implications.

This property of labels can be also employed in determining the valid comparisons between attribute classes. In fact, the attributes of a class can be distinguished into three main categories:

1 the attributes originally defined in this class ;

2. the inherited yet refined attributes ; and

3. the inherited but not refined attributes.

Assuming the following one to one mappings:

i. o1: I --> O

ii. o2: L --> O

iii. id1: L --> I

where O is the universal set of objects, I is the set of the system identifiers and L is the set of the logical names, we can define these attribute categories as follows:

Definition 1: The *Modifier* of a class C, M[C] is defined as the set of the labels of all the attributes included in its definition:

$$M[C]=\left\{ x \mid id1(C.x)\ \varepsilon\ o2(C).A \right\}$$

where o.A is the set of the system identifiers for the attributes of the object o

The term modifier has been introduced in[39].

Definition 2: The set of the *unordered superclasses* of a class C, US[C]is defined as:

$$US[C]=$$
$$\left\{ x \mid (id1(x)\ \varepsilon\ o2(C).isa^*) \text{ and } (not\ (exists\ y : (id1(y)\ \varepsilon\ o2(C).isa^*) \text{ and } (id1(x)\ \varepsilon\ o2(y).isa^*))) \right\}$$

where o.isa* is the transitive closure of the superclasses of the object o.

Definition 3: The *Intension* of a class C, INT[C] is defined as:

$$a.\ INT[C]=M[C] \text{ if } US[C]\ is\ empty$$
$$b.\ INT[C]=\left\{ x \mid (x\ \varepsilon\ M[C])\ or\ (x\ \varepsilon \bigcup_{j\ \varepsilon\ US[C]} INT[j]) \right\} otherwise$$

Definition 4: The *Horizontal Extension* of a Class C, with respect to a subset S={S1,...,Sm} of its superclasses, HE[C,S] is defined as:

$$HE[C,S]=$$
$$\left\{ x \mid (x\ \varepsilon\ INT[C]) \text{ and } ((not(x\ \varepsilon \bigcup_{i\ \varepsilon\ S} Int[i])))or(Forall\ i\ in\ S:(x\ \varepsilon\ INT[i])\rightarrow(not\ (id1(i.x)\ \varepsilon\ o2(C.x)isa^*)))) \right\}$$

Thus, the horizontal extension of a class C with respect to some of its superclasses S1,...,Sm includes the labels of the attribute classes of C, which are not associated with any attribute classes of S1,...,Sm, if they are superclasses of the former(see a somehow different definition in [37]).

Definition 5: The *Vertical Replacement* of a class C, with respect to any of its superclasses S, VR[C,S] is defined as:

$$VR[C,S] = \left\{ x \mid (x \ \varepsilon\ INT[C] \text{ and } (x \ \varepsilon\ INT[S]) \text{ and } (id1(S.x)\ \varepsilon\ o2(C.x).isa^*) \right\}$$

In words, the vertical replacement of a class with respect to one of its superclasses, includes the labels of its attributes which also specialize attributes with the same label inherited from this superclass.

Definition 6: The *Comparison Basis for Attribute Classes* of two classes C1 and C2, CBAC[C1,C2] is defined as:

$$CBAC[C1,C2] =$$

$$\left\{ x \mid (id1(x)\ \varepsilon\ o2[C1].isa^*) \text{ and } (id1(x)\ \varepsilon\ o2[C2].isa^*)\ and\,(\ not\ \ (exists\ \ y: \right.$$

$$\left. (id1(y)\ \varepsilon\ o2[C1].isa^*) \text{ and } (id1(y)\ \varepsilon\ o2[C2].isa^*) \text{ and } (id1(x)\ \varepsilon\ o2(y).isa^*))) \right\}$$

Since the Isa-relation is only a partial ordering and a class may have more than one superclasses, two classes C1 and C2 in general may have more than one pairwise unordered minimal common superclases. These will be the elements of their Comparison Basis for attribute classes(CBAC set). The CBAC set is guaranteed to be non-empty, since the most general TELOS built-in class at each level of classification is a superclass of all the classes at that level(see figure 1).

Given the previous definitions, we can distinguish between two categories of attributes that can be compared, with respect to two classes and their Comparison Basis:

1. The *Common Vertical Replacement*, which is a set of attribute pairs (c1.x, c2.y) defined as:

$$CVR[c1,c2,S] = \bigcup_{si\ \varepsilon\ CBAC[c1,c2]} CVR[c1,c2,Si]$$

where

$$CVR[c1,c2,Si] = \left\{ (c1.x,\ c2.y) \mid (x\ \varepsilon\ VR[c1,Si]) \text{ and } (y\ \varepsilon\ VR[c2,Si]) \text{ and } (x = y) \right\}$$

The common vertical replacement includes the commonly inherited attributes, which have also been refined within the classes of consideration.

2. The *Unique Vertical Replacements* UVR[c1,c2,Si] and UVR[c2,c1,Si],are defined with respect to each of the superclasses Si in the comparison basis of the classes c1,c2 according to the following definition:

$$UVR[x1,x2,Si]=\left\{(x1.x,\ Si.y)\,|\,(x\ \varepsilon\ VR[x1,Si])\text{ and }(not\ (x\ \varepsilon\ VR[x2,Si]))\text{ and }(x=y)\right\}$$

These sets pair attributes commonly inherited by superclasses, having identical labels but which are specialized in exactly one of the classes in hand.

The union of the Common and the Unique Vertical Replacements, defined as:

$$AT[C1,C2]=$$

$$CVR[C1,C2,CBAC[C1,C2]]\bigcup_{Si\ \varepsilon\ CBAC[C1,C2]}\left\{UVR[C1,C2,Si]\right\}\bigcup_{Si\ \varepsilon\ CBAC[C1,C2]}\left\{UVR[C2,C1,Si]\right\}\quad(1)$$

includes the pairs of attribute classes that must be taken into account in the estimation of the distance of two objects with respect to the aggregation abstraction. Note that attribute classes that belong to the horizontal extensions of the involved classes are not further compared, since they are not applicable to both of them(see the principle of the partially uniform representation).

5.2. Comparisons Between Attribute Tokens

Since Isa relations are not defined for attribute tokens and their labels serve only as references within the relevant object-scopes but are meaningless outside them(e.g "mike_car" label in figure 4). the comparison pairs of attribute tokens can not be formed as in the case of the attribute classes.

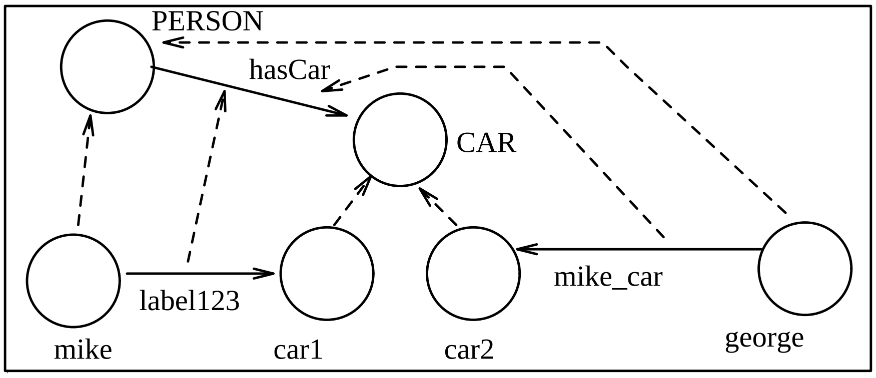

Fig. 4. Entity objects and attribute tokens

Actually, the semantics of attribute tokens are expressed by their attribute classes. Therefore, their semantic identity can be detected from the labels of these classes. For instance, in the 4th figure we can compare the attribute token *mike.mike_car* with the attribute token *george.label123* due to their common classification under the attribute class *Person.hasCar* and despite their different labels.

In conclusion, attribute tokens yield comparison-pairs as follows:

Initially, the comparison basis for attribute tokens(CBAT) of two objects Oi, Oj is defined as:

$$CBAT[Oi,Oj]=Oi.in \cap Oj.in \text{ if } this\ intersection\ is\ not\ empty \quad otherwise,$$

$$CBAT[Oi,Oj]=$$

$$\left\{x \mid (id1(x)\ \varepsilon\ CC(Oi,Oj)) \text{ and } (not\ (exists\ y:(id1(y)\ \varepsilon\ CC(Oi,Oj)) \text{ and } (id1(x)\ \varepsilon\ o2(y).isa*)))\right\}$$

where CC(Oi,Oj) is the set of the common classes of Oi, Oj, defined as:

$$CC(Oi,Oj)= Oi.in* \cap Oj.in*$$

where:

$$Ok.in*=\bigcup_{i\ \varepsilon\ Ok.in} i.isa*$$

Then the set of the attribute classes which are applicable to both Oi and Oj(CAC) is defined as:

$$CAC[Oi,Oj]=\bigcup_{i\ \varepsilon\ CBAT[Oi,Oj]} INT[i]$$

Finally, we define the comparison-pairs consisting of sets of attribute tokens as:

$$CAT[Ok,Or]=$$

$$\left\{(\left\{X1,...,Xm\right\},\left\{Y1,...,Yn\right\}) \mid (forall\ i,j:(id1(Xi)\ \varepsilon\ Ok.A) \text{ and } (id1(Yj)\ \varepsilon\ Or.A) \text{ and }\right.$$

$$\left.(exists\ z:(z\ \varepsilon\ CAC[Ok,Or]) \text{ and } (id1(z)\ \varepsilon\ o2(Ok.Xi).in*) \text{ and } (id1(z)\ \varepsilon\ o2(Or.Yj).in*)))\right\} (2)$$

This definition reflects the possibility of objects instantiating relevant attribute classes by more than one attribute tokens.

5.3. An Example of Attribute Comparisons

According to the schema of the figure 5 we can form the following sets with respect to the classes UniversityDoctor(ud) and Professor(p):

CBAC[ud,p] = { AcademicStaff }

HE[p,AcademicStaff] = { directs }

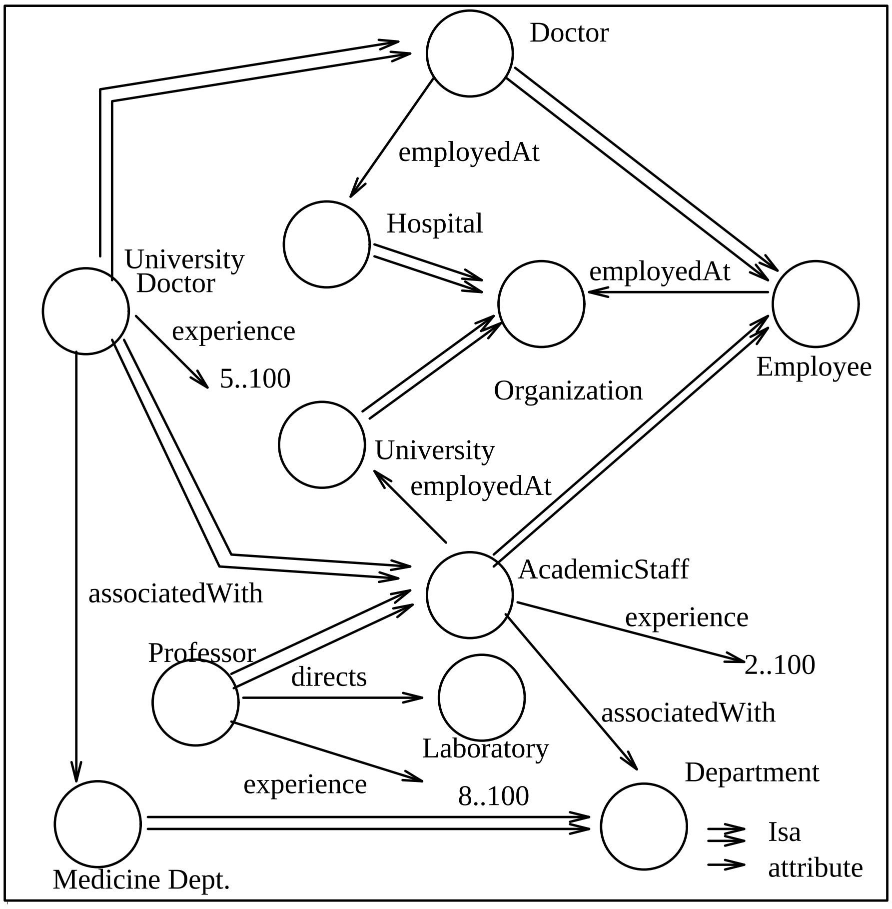

Fig. 5. A schema for comparison of attribute classes

HE[ud,AcademicStaff] = { employedAt FROM doctor }

VR[ud,AcademicStaff] = { associatedWith, experience }

VR[p,AcademicStaff] = { experience }

CVR[p,ud,{AcademicStaff}] = {(p.experience, ud.experience)}

UVR[ud,p,AcademicStaff] = {(ud.associatedWith,AcademicStaff.associatedWith)}

UVR[p,ud,AcademicStaff] = {}

Note that when two attributes that have the same label are inherited from two distinct superclasses, they must be disambiguated with a FROM clause within the scope of the inheriting class, as in the case of the class Professor and the attribute employedAt.

Also, in the schema of the figure 6 the comparison pairs of the attribute tokens are:

CBAT[george, kate]={employee}

CAC[george,kate]={employee.salary, employee.experience}

CAT[george,kate]={({george.salary1},{kate.salary123,kate.salary9}), ({george.label11},{kate.experience})}

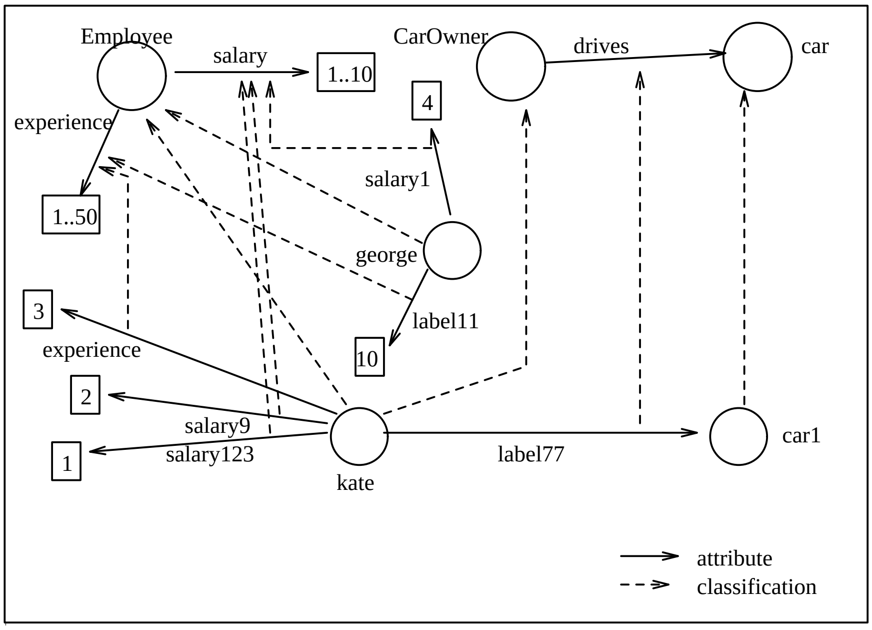

Fig. 6. A schema for comparisons of attribute tokens

Thus, george and kate are not comparable in terms of car ownership since the relevant attributes are not applicable to both of them.

6. General Forms of Distance Metrics

According to the previous analysis, we can propose four general functional forms for distance metrics. These forms correspond to entity classes, attribute classes, entity tokens and attribute tokens.

i. Entity classes,

F2d(C1,C2)=f(d1(Id(c1),Id(c2)),d2(In(c1),In(c2)),d3(Isa(c1),Isa(c2)),d4(AC’),d5(AT’))

ii. Attribute classes,

D(ac1,ac2)=G(d1(Id(ac1),Id(ac2)),d1(Id(From(ac1)),Id(From(ac2))),d2(In(ac1),In(ac2)), d3(Isa(ac1),Isa(ac2)),d4(AC’),d5(AT’),D(o1(Id(To(ac1))),o1(Id(To(ac2)))))

iii. Entity tokens,

D(t1,t2)=F'(d1(Id(t1),Id(t2)), d2(In(t1),In(t2)), d5(AT'))

iv. Attribute Tokens,

D(at1,at2)=G'(d1(Id(at1),Id(at2)),d1(Id(From(at1)),Id(From(at2))), d2(In(at1),In(at2)),d5(AT'),D(o1(Id(To(at1))),o1(Id(To(at2)))))

where

* *F,F'*	are aggregate functions measuring the overall distances of entities
* *G,G*	are aggregate functions measuring the overall distances of attributes
* *d1*	is the identity function over the system identifiers
* *d2*	denotes a distance metric over the classification abstraction
* *d3*	denotes a distance metric over the generalization abstraction
* *d4,d5*	are aggregate functions reflecting distances over the attribution abstraction
* *AC'*	is the set of the corresponding attribute classes for the objects in hand defined according to formula (1)
* *AT'*	is the set of the corresponding attribute tokens for the objects in hand defined according to formula (2)

These general functional forms can be viewed as abstractions of various comparison methods proposed for analogical reasoning and specialized in at least the following ways or combinations of them:

i. the contribution of the partial distances to the overall measure ;

ii. the degree of recursiveness of the partial distances d2, d3, d4 and d5 ; and

iii. the salience associated with the elementary feature-distances of the aggregation dimension.

We believe that such a specialization must take into account a pragmatic consideration suggesting a trade off between the quality of the final estimate and the computational cost for obtaining it.

7. Conclusions and Issues For Further Research

In this paper we presented a qualitative framework for computing similarity, within the context of analogical software reuse. We argued that similarity measures must be devised according to a set of principles, distilled from requirements of the analogical reasoning process.

We also developed a systematic theoretical comparison basis and a general distance model for similarity computation over a particular representational notation for objects.

Further research aims at:

* the precise definition of partial distance metrics according to the semantics of the relevant abstractions,

* the contribution of these metrics to the overall similarity estimate,

* the quantification of the salience of the various object attributes,

* the extension of the comparison basis of the attribute classes in a way dealing with the synonyms/homonyms problem[3], by exploiting the classification of attribute classes into common metaclasses, and,

* issues of computational efficiency.

A prerequisite for the integration of an adequately instantiated distance model into tools supporting analogical software reuse, will be its experimental validation in a relevant context. Such an experimental validation is to be attempted against existing repositories of descriptions of software artifacts including the Software Information Base developed in the ITHACA project [7].

References

1. Bareiss R., King J., Similarity Assessment in Case-Based Reasoning, DARPRA Workshop on Case-Based Reasoning, 1988

2. Bergadano F., et al., Learning Two-Tiered Descriptions of Flexible Concepts: The Poseidon System, Machine Learning 8, 1988

3. Bhargava H. et al., Unique Names Violations, a Problem for Model Integration or You Say Tomato, I Say Tomahto, ORSA Journal on Computing 3(2), 1991

4. Biggerstaff T., Richter C., Reusability Framework, Assessment and Directions IEEE Software, March 1987

5. Burton B. et al., The Reusable Software Library, IEEE Software, July 1987

6. Cohen P., Kjeldsen R., Information Retrieval by Constrained Spreading Activation in Semantic Networks, Information Processing and Management, 23(4), 1987

7. Constantopoulos P., et al., The Software Information Base: A Server for Reuse, Technical Report, Institute of Computer Science, Foundation of Research and Technology-Hellas, February 1993

8. Curtis B., Cognitive Issues in Reusing Software Artifacts, Software Reusability, Addisson-Wesley, 1987

9. Pietro-Diaz R., Freeman P., Classifying Software for Reusability, IEEE Software, January 1987

10. Esposito F. et. al. Classification in Noisy Environments Using a Distance Measure

Between Structural Symboic Descriptions, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(3), 1992

11. Fischer G., Cognitive View of Reuse and Redesign, IEEE Software, July 1987

12. Gennari J., et al., Models Of Incremental Concept Formation, Artificial Intelligence 40, 11-61, 1989

13. Gentner D., Structure-Mapping: A Theoretical Framework for Analogy, Cognitive Science 7, 1983

14. Gentner D., Analogical Inference and Analogical Access, Analogica, Armand Prieditis(ed.), Morgan Kaufmann Pub., 1988

15. Gentner D., Finding the Needle: Accessing and Reasoning From Prior Cases, DARPRA Workshop on Case-Based Reasoning, 1988

16. Golding A., Rosenbloom P., Combining Analytical and Similarity-Based CBR, DARPRA Workshop on Case-Based Reasoning, 1988

17. Greiner R., Abstraction-Based Analogical Inference, Analogical Reasoning, Kluwer Academic Publishers, 1988

18. Hall R., Computational Approaches to Analogical Reasoning: A Comparative Analysis, Artificial Intelligence 39, 1989

19. Holyoak K., Koh K., Surface and Structural Similarity in Analogical Transfer, Memory and Cognition 15(4), 1987

20. Kolonder J., Judging Which is the "Best" Case for a Case-Based Reasoner, DARPRA Workshop on Case-Based Reasoning, 1988

21. Kowalski H.J., Topological Spaces, Academic Press, 1965

22. Leite J, Freeman P., Requirements Validation Through Viewpoint Resolution, IEEE Transactions on Software Engineering 17(12), 1991

23. Maiden N., Sutcliffe A., Exploiting Reusable Specifications through Analogy, Communications of the ACM, 35(4), 1992

24. Maiden N., Sutcliffe A., Analogical Matching for Specification Reuse, Proceedings of the 6th Annual Conference on Knowledge-Based Software Engineering, IEEE Computer Society Press, 1991

25. Michalski R., Learning from Observation: Conceptual Clustering, Machine Learning: an AI approach, Vol I, Morgan Kaufmann Pub., 1986

26. Miriyala k., Harandi M., The role of analogy in Specification Derivation, Proceedings of the 6th Annual Conference on Knowledge-Based Software Engineering, IEEE Computer Society Press, 1991

27. Mylopoulos J., et. al., Telos: Representing Knowledge About Information Systems, ACM Transactions on Information Systems, 8(4), 1990

28. Novick L., Analogical Transfer: Processes and Individual Differences, Analogical Reasoning, Kluwer Academic Publishers, 1988

29. Rosch E., et all., Basic Objects in Natural Categories, Gognitive Psychology 8, 1976

30. Russel S., Analogy By Similarity, Analogical Reasoning, Kluwer Academic Publishers, 1988

31. Schwanke R., An Intelligent Tool for Re-engineering Software Modularity, ICSE-13, Austin Texas, 1991

32. Seifert C., Hammond K., Why There Is No Analogical Transfer, DARPRA Workshop on Case-Based Reasoning, 1988

33. Sjoberg L., A Cognitive Theory of Similarity, Goteborg Psychological Reports, Number 10, Volume 2, 1972

34. Smith E.E., Concepts and Induction, Foundations of Cognitive Science, A Bradford Book, The MIT Press, 1989

35. Step R., Michalski R., Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects, Machine Learning: an AI approach, Morgan Kaufmann Pub., 1986

36. Tversky A., Features of Similarity, Psychological Review, 44(4), July 1977

37. Vassiliou Y. et. al., Technical Description of the Software Information Base, ITHACA.FORTH.91.E2.#2,1990

38. Wegner P., The Object-Oriented Classification Paradigm, Research Directions in Object-Oriented Programming, ed. Shriver, Wegner, The MIT Press

39. Wegner P., Zdonic S., Inheritance as an Incremental Modification Mechanism or What Like is and Isn't Like, Proceedings of the European Conference on Object-Oriented Programming(ECOOP'88), Lecture Notes in Computer Science, 1988

40. Winston P., Learning and Reasoning by Analogy, Communications of the ACM, 23(12), December 1980

41. Wrobel S., Concept Formation in Man and Machine: Fundamental Issues, Workshop

on Concept Formation in Man and Machine, GMD, 1991