Measuring Similarity Between Software Artifacts

George Spanoudakis
Panos Constantopoulos
Department of Computer Science
University of Crete
and
Institute of Computer Science
Foundation of Research and Technology, Hellas

Abstract

This paper presents a model of estimating the similarity of software artifacts so as to promote their analogical reuse. The model permits comparisons between artifacts developed at the various stages of the software development (i.e specifications, designs and code) from conceptual descriptions of these artifacts. This is achieved by using metrics measuring the distance between such descriptions with respect to general conceptual modeling abstractions(i.e the classification, the generalization and the attribution) underlying them[21]. Similarity estimates are influenced by measures of salience of the involved attributes. Salience is measured as belief on three domain independent properties of attributes(i.e the characticity, the abstractness and the causality), suggested as predictive of their significance[22]. A prototype of the model is presented together with an example of using similarity to support the specification of requirements by reuse.

1. Introduction

Reuse has been realized as a form of analogical problem solving[12,13] where knowledge inherent in artifacts of existing software systems(*source artifacts*) is transferred to artifacts of a new system under development(*target artifacts*) on the basis of identifying analogies between them.

Analogies may be utilized in both retrieving and comprehending artifacts. Retrieval is supported by analogical inexact matches between artifacts. Also, comprehensibility is improved by identifying analogical correspondences between possibly not well understood features of existing artifacts and well understood features of target artifacts, which reflect directly the mental models of the software engineers, who invented them.

In this paper, we present a model for estimating the analogical similarity between software artifacts of any type(i.e code, designs or specifications) so as to promote their *compositional* reuse[5].

Similarity is measured from conceptual descriptions of artifacts. It is not based on criteria valid only to artifacts of a particular substance(e.g the *subsumption relation* between code artifacts in [16], the information hiding in [27]) or on domain specific representation constructs or on explicitly asserted distance measures(e.g weighted facet-terms graphs in [6,16]). It is rather based on partial metrics, which measure the distance between descriptions with respect to general semantic modeling abstractions underlying them. These abstractions include the *classification*, the *generalization* and the *attribution*. As it has been pointed out elsewhere[3], these abstractions comprise a representational framework adequate for describing artifacts, generated in all the stages of the software development. Thus, we can compare requirements specifications, designs or code regardless of the model of their descriptions, provided that this model can be defined through the mentioned general abstractions.

The proposed similarity model exhibits properties observable in the assessment of inter-object resemblances by humans[20,25,28]. Furthermore, it measures the salience of different description elements in an attempt to decrease the misleading role of superficial elements in useless or inconsistent transfers of knowledge between descriptions, evident in analogical reasoning[7,15] and software reuse[12].

The rest of the paper is organized as follows. In section 2, we introduce the TELOS language which is the conceptual modeling framework of the similarity model. In section 3, we describe the metrics comprising the model, defined in detail in [21]. In section 4, we describe metrics measuring the salience of different description elements. In section 5, we present a prototype system incorporating the similarity model, and in section 6, we present an example on how similarity can support the specification of requirements by reuse. Finally in section 7, we summarize the basic aspects of the approach together with the issues open to further research.

2. TELOS: A Conceptual Modeling Language Adequate for Similarity Computation

As it has been pointed out in[3] a modeling formalism adequate for describing artifacts of the S/W life cycle, must be expressive enough to support at least: (i) complex, nested and recursive structures; (ii) multiple or hierarchical organizations of entities; (iii) typed and usually attributed relationships and (iv) grouping into multi-

ple and possibly overlapping spaces providing different views on entities.

These requirements motivated the choice of the TELOS language as the conceptual modeling framework of the Similarity Model. TELOS has been acknowledged as a language adequate for describing software artifacts elsewhere[3,14] due to its object-oriented data model, which offers powerful modeling abstractions (i.e *multiple and meta classification*, *multiple generalization* and *typed attribution*). Although, the interested reader may refer to [10] for a formal presentation of the language, in this section, we define in a set theoretic manner some aspects of TELOS, necessary for the subsequent presentation of the Similarity Model.

TELOS objects are partitioned according to their *classification level*(i.e Tokens and Classes further partitioned into Simple Classes, Meta Classes, Meta Meta Classes and so on) and their *role* (i.e Attributes vs. Entities).

On the basis of these partitions we can distinguish four basic categories of TELOS objects having the following tuple forms:

Entity Tokens $(E_t) : O_i=[i,In,A]$

Entity Classes $(E_c) : O_i=[i,In,Isa,A]$

Attribute Tokens $(A_t) : O_i=[i,FROM,In,A,TO]$

Attribute Classes (A_c): $O_i=[i,FROM,In,Isa,A,TO]$

In these forms, i is a system identifier uniquely identifying the object O_i, *In* is a set of system identifiers denoting the direct classes of object O_i(i.e the classes in which an object has been explicitly classified), *Isa* is a set of system identifiers denoting the direct superclasses of object O_i (i.e the classes which have been explicitly declared as the superclasses of it), *A* is a set of system identifiers denoting the direct attributes(i.e those not inherited) of object O_i, *FROM* is the identifier of the object owning the attribute O_i and *TO* is the identifier of the object pointed to by the attribute O_i(i.e its class range).

TELOS objects also have *logical names*, which are unique to them in the case of entity objects and possibly shared by more than one attribute objects. Each TELOS class i has an *intension INT[i]*(i.e a set including the identifiers of the attributes it introduces, inherits or refines) and an *extension EXT[i]*(i.e a set including the identifiers of its instances). Three more concepts about TELOS attribute classes, namely their *scopes*, their *refining classes* and their *possible ranges* are necessary for the definition of salience. Assuming an 1-1 mapping $l : I \rightarrow L$ where I, L are the sets of the system identifiers and the logical names of all the objects in a TELOS object base, these three concepts are defined as follows:

1. The scope of an attribute i, S[i](i.e the classes to which it applies) is defined as:

$$S[i] = \{ c \mid (exists\ x,\ j : (x = o_i.FROM) \text{ and } (c\ \varepsilon\ o_x.isa^*) \text{ and } (j\ \varepsilon\ INT[c]) \text{ and } (l(i) = l(j))\ or\ (o_i.FROM\ \varepsilon\ o_c.isa^*)) \}$$

$O_x.isa^*$ is the transitive closure of the superclasses of O_x.

2. The set of the refining classes of an attribute i, R[i](i.e the classes which inherit and refine i) is defined as:

$$R[i] = \{ c \mid (c\ \varepsilon\ S[i]) \text{ and } (exists\ x : (x\ \varepsilon\ o_c.A) \text{ and } (l(i) = l(x))) \}$$

3. The set of the possible attribute ranges of an attribute i(i.e the classes serving as its ranges), AR[i], is defined as:

$$AR[i] = \{ x \mid (exists\ y,z : (y\ \varepsilon\ R[i]) \text{ and } (z\ \varepsilon\ o_y.A) \text{ and } (l(z) = l(i)) \text{ and } (o_z.TO = x)) \}$$

3. Estimation of Similarity

In the following, we present the partial distance metrics of the Similarity Model. These metrics are: (i) the identity distance d_1; (ii) the classification distance d_2; (iii) the generalization distance d_3 and, (iv) the attribution distance d_4

Distance over the Identifiers. The metric d_1 defined as the identity function:

$$d_1(O_i, O_j) = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i \equiv j \end{cases}$$

distinguishes between identical and non identical objects.

Distances over the Classifications and the Generalizations of Objects. The classification and generalization distances are defined as:

$$d_2(O_i.In, O_j.In) = \sum_{x\ \varepsilon\ \{O_i.in^* - O_j.in^*\} \cup \{O_j.in^* - O_i.in^*\}} \frac{1}{D_x}$$

$$d_3(O_i.Isa, O_j.Isa) = \sum_{x\ \varepsilon\ (O_i.isa^* - O_j.isa^*) \cup (O_j.isa^* - O_i.isa^*)} \frac{1}{D_x}$$

$O_i.in^*$ is the union of the transitive closures of the superclasses of the classes of object i and D_x is the depth of a class with respect to the Isa-graph of its classification level.

Functions d_2 and d_3 approximate the number of the non uniform attributes of two objects(see relevant criteria below), in the absence of any explicit information about them. In TELOS, an object can instantiate classes of attributes defined in its classes, while it inherits(and possibly refines) attributes defined in its superclasses. Thus, the classes and the superclasses which are unique to only one

of two objects, are the sources of non uniform attributes of these objects. Therefore, the number of these classes approximates the number of the non uniform attributes of the involved objects. To improve further the quality of this approximation, functions d_2 and d_3 exploit a hypothesis empirically verified in [18]. According to it, the deeper a class in an Isa hierarchy the less the attributes it introduces. Consequently, functions d_2 and d_3 weight the non common classes and superclasses by the inverse of their depths in the Isa-graphs.

Distance over the attribution of Objects. Due to the plethora of the models that have been proposed for describing software artifacts and their different application domains, we cannot assume a single predefined set of attributes for describing them. Therefore it is necessary to find criteria for deciding whether attributes in different descriptions could be semantically homogeneous or not.

We adopt two criteria of semantic homogeneity and preclude comparisons between attributes that do not satisfy these criteria, which are not semantically interpretable.

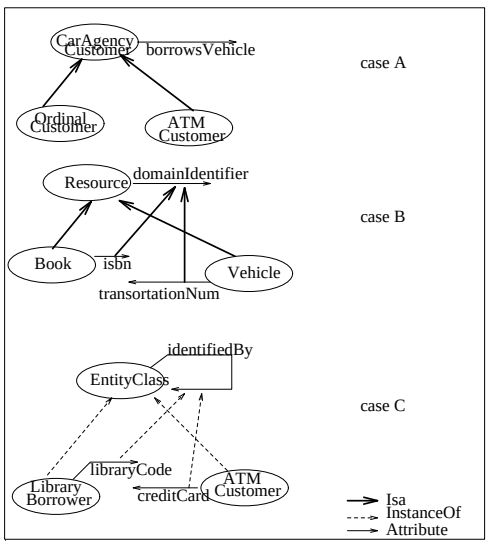

Figure 1: Semantically homogeneous attributes

The first criterion is the sharing of an attribute superclass having a logical name identical to the logical names of the involved attributes (e.g the inherited attribute *borrowsVehicle* of case A in figure 1). According to this criterion we define the set of pairs of the *common attributes* of two objects as:

$$S_1(O_i,O_j)=\{(x,y)\,|\,x \in INT[i] \text{ and } y \in INT[j] \text{ and } (exists\ z:$$

$$(z \in o_x.isa^*) \text{ and } (z \in o_y.isa^*) \text{ and } (l(x)=l(y)) \text{ and } (l(x)=l(z)))\}$$

The second criterion is the sharing of at least one common attribute superclass or at least one common attribute metaclass. For instance, the attribute classes *isbn* of *Books* and *transportationNum* of *Vehicles*, are semantically homogeneous since they both specialize the general attribute class *domainIdentifier* of *Resources*(case B of figure 1). Similarly, the attributes *libarryCode* and *creditCard* are semantically homogeneous since they both instantiate the attribute metaclass *identifiedBy*, which groups all the attributes serving as identifiers of entity classes(e.g *LibrarryBorrower*) as it is presented in case C of figure 1. According to this criterion we define the set of pairs of the *semantically homogeneous* attributes of two objects, as:

$$S_2(O_i,O_j)=\{(x,y)\,|\,x \in INT[i] \text{ and } y \in INT[j] \text{ and } (exists\ z:$$

$$((z \in o_x.isa^*) \text{ and } (z \in o_y.isa^*)) \text{ or } ((z \in o_x.in^*) \text{ and } (z \in o_y.in^*)) \text{ and}$$

$$(not\,(exists\ (u,w): ((u,w) \in S_1(O_i,O_j)) \text{ and } ((x=u) \text{ or } (y=w))))\}$$

The attributes of two objects which do not satisfy the previous criteria form two additional sets:

i. The set of the *non uniform* attributes of object Oi, with respect to object Oj, defined as:

$$S_3(O_i,O_j)=\{x\,|\,x \in INT[i] \text{ and } (not\,(exists\ (u,w):$$

$$(((u,w) \in S_1(O_i,O_j)) \text{ or } ((u,w) \in S_2(O_i,O_j))) \text{ and } (x=u)))\}$$

ii. The set of the *non uniform* attributes of object Oj, with respect to object Oi, defined as:

$$S_4(O_i,O_j)=\{x\,|\,x \in INT[j] \text{ and } (not\,(exists\ (u,w):$$

$$(((u,w) \in S_1(O_i,O_j)) \text{ or } ((u,w) \in S_2(O_i,O_j))) \text{ and } (x=w)))\}$$

According to these criteria of homogeneity, the distance over the attribution of two objects d_4 is defined as:

$$d_4(O_i.A,O_j.A) = f_1(S_1(O_i,O_j)) + f_2(S_2(O_i,O_j)) +$$

$$f_3(S_3(O_i,O_j)) + f_3(S_4(O_i,O_j))$$

f_1 measures the distance over the *common attributes* of objects Oi and Oj and is defined as:

$$f_1(S_1(O_i,O_j)) = \sum_{(x,y) \in S_1(O_i,O_j)} s(x)s(y)D(O_x,O_y)$$

f_2 measures the distance according to the *semantically homogeneous attributes* of two objects. It considers only 1-1 mappings between attributes, since humans attempt to interpret analogies between objects in terms of such mappings [7]. Given all the possible 1-1 mappings between the attributes of two objects, defined as:

$$C[O_i,O_j]=\{c_k\,|\,(c_k \subseteq S_2(O_i,O_j)) \text{ and } (forall\ (x,y):$$

$$((x,y) \in c_k) \rightarrow (not\,(exists\ (u,w):((u,w) \in c_k) \text{ and}$$

$$(((x \equiv u) \text{ and } (y \neq w)) \text{ or } ((x \neq u) \text{ and } (y \equiv w))))))\}$$

function f_2 selects the mapping with the minimum overall distance, according to the following definition:

$$f_2(S_2(O_i,O_j)) = \min_{c_k \varepsilon C[O_i,O_j]} \left\{ \sum_{(x,y) \varepsilon c_k} s(x)s(y)D(O_x,O_y) + \sum_{u \varepsilon A_i[c_k]} s(u)^2 + \sum_{w \varepsilon A_j[c_k]} s(w)^2 \right\}$$

$A_i[c_k]$ ($A_j[c_k]$) are the attributes of object i(j), which are mapped onto no attribute of object j(i) given the combination C_k. Formally, these sets are defined as:

$$A_m[c_k] = \{x \mid (x \varepsilon INT[m]) \text{ and } (not(exists\ (u,w): ((u,w) \varepsilon c_k) \text{ and } (x \equiv u)))\} \; m=i,j$$

f_3 measures the distance over the non uniform attributes of two objects and is defined as:

$$f_3(S_u(O_i,O_j)) = \sum_{x \varepsilon S_u(O_i,O_j)} s(x)^2 \quad (u=3,4)$$

In the definitions of the functions f_1 and f_2, D is the aggregate distance defined in the following. Also the s(i) factors refer to the salience of the relevant attributes, which is estimated by the belief functions outlined in section 4.

The Aggregate Distance Metric D. The presented partial distances are aggregated into an overall distance D according to the following linear combination:

$$D(O_i,O_j) = w_1 * d_1(O_i,O_j) + w_2 * d_2(O_i.In, O_j.In) + w_3 * d_3(O_i.Isa, O_j.Isa) + w_4 * d_4(O_i.A, O_j.A))$$

This overall distance is transformed into a similarity measure, according to the formula: $f(D) = e^{-N*D}$ justified by evidence about the way that humans react to observed differences between objects in assessing their similarity[19]. The parameter N determines the sensitivity of similarity measures w.r.t distance measures.

4. Estimating the Salience of Attributes

Salience distinctions are critical for analogical reasoning [7]. Salient features are used by experts in retrieving source descriptions analogous to a given target[Hol88]. Furthermore, only analogies between them may lead to consistent and pragmatically useful inferences of knowledge about the target, from the retrieved sources. On the contrary, non salient features may only need to inconsistent or useless analogical transfers but nevertheless they facilitate retrieval as far as novices are concerned[15].

Estimates of salience in general computational models of analogical reasoning usually depend on explicitly asserted knowledge [4,17,26] or judgements of the results of some reasoning session obtained from some external observer[1]. As such, they are knowledge acquisition intensive or sensitive to subjective personal judgements.

As opposed to such estimates, we realize salience as belief that an attribute is dominant and define it as a function:

$S: A_c \rightarrow [0,...,1]$ such that forall i

$(i \varepsilon A_c) \rightarrow S(i) \varepsilon [Bel(DOM(i)),...,P^*(DOM(i))]$

where A_c is the set of TELOS attribute classes $Bel(DOM(i))$ is the belief on the dominance of attribute i and $P^*(DOM(i))$ is the upper probability of the dominance of attribute i as they are defined by the Dempster-Shafer theory of evidence[24]

The dominance of an attribute i, DOM(i) is defined in terms of three other properties of it, namely the *characticity(CHi)*, the *abstractness(ABSi)* and the *causality(CSLi)* as: (ABS_i and CH_i) *or* CSL_i

In the following, we discuss these properties and present belief measures about them, defined over conceptual schemas.

Judging Belief about Characticity of Attributes. According to [22] characteristic are attributes which have distinct range classes in all the classes, which inherit them. The specialization of the class range of such attributes in certain subclasses expresses how these subclasses differ from their superclasses and in this sense characterizes them. We define the belief to the characticity of an attribute i as:

$$Bel(CH_i) = c_i = \frac{|AR[i]|}{|S[i]|} \text{ and } Bel(\overline{CH_i}) = 1 - c_i$$

According to this measure, the more the classes that refine an inherited attribute(see the definition of the attribute scope in section 2) by associating it with a distinct range class, the more characteristic this attribute is for the classes of its scope.

Figure 2 presents an example of more and less characteristic attributes in an Isa taxonomy of object types in object oriented programming languages modeled according to [29].

In this taxonomy the attribute *method* expressing the methods supported by an object type is very characteristic(Bel=0.8), since it has distinct range classes in most of the classes of its scope. Unlike it the attribute *author* has a low characticity (Bel=0.2). This attribute has been modeled in a way implying that nothing differentiates between software engineers, who implement CooL, Eiffel or C++ object types, worth of being abstracted into the conceptual schema.

Judging Belief about Abstractness of Attributes. The abstractness of an attribute reflects how essential it is for the existence or the behavior of the objects having it. This

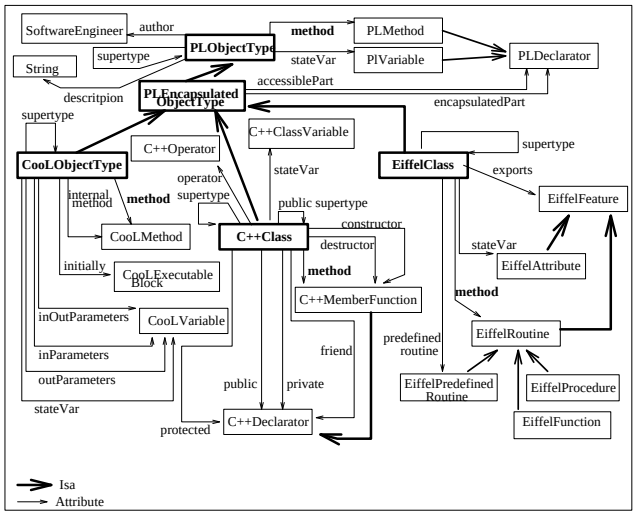

Figure 2: Object Types of OO Languages

property depends on kind of the class, which introduces an attribute in a conceptual schema[22]. Classes may be distinguished into *abstract* (i.e classes having no instances of their own[9,11,23,30]) and *concrete*(i.e classes having their own instances). Abstract classes tend to introduce attributes essential for the composition and the functionality of their instances[18]. We define the abstractness of an attribute i as belief that its introducing class j is abstract:

$$Bel(ABS_i) = e_j = \frac{|EXT_s[j]|}{|EXT[j]|} \text{ and } Bel(\overline{ABS_i}) = 1 - e_j$$

Thus the more the instances in the extension of a class, that also belong to some other class, which is not a superclass of it, the higher the belief to the abstractness of both this class and the attributes it introduces.

In figure 2, the class *PLObjectType* has no instances of its own(any object type must be implemented in some specific programming language). Consequently, the belief to the abstractness of the attributes it introduces(i.e *method*, *stateVar*, *supertype*, *and author*) is 1. Unlike these attributes, the attribute *initially* of the class *CooLObjectType*, which in turn has no shared instances, has an abstractness of 0. In fact, this attribute expresses a special construct of the CooL programming language for incorporating code initializing object types, which is not offered by all the object oriented programming languages and thus it is not essential for object types in general.

Judging Belief about Non Causality of Attributes. We consider as *causal*, attributes whose values, determine the values of other attributes and as such they are important in inferencing these latter values. For example, the model of a car could dictate the place where it has been produced, if the relevant manufacturer produces its different models, in different countries. This notion of causality has also been realized in the area of explanation based learning, where predefined types of dependency relations are offered in languages for specifying explanations [17].

In order to avoid the cost of acquiring such relations, we restrict our interest in dependencies between attributes that have the same domain as well as in a special case of them, the *total equivalences*. Both these dependencies have the special property that the conceptual schema may imply their impossibility.

Formally, if two attributes x and y are defined as associations of the form: $x:D_x \rightarrow I_x \quad y:D_y \rightarrow I_y$ where I_x and I_y denote the images(i.e the sets of their actual values) of the attributes x and y, we will say that the attribute y depends on the attribute x(i.e *dep(x,y)*) if and only if they have identical domains(i.e $D_x \equiv D_y$). A dependency dep(x,y) can be defined as a mapping $M:I_x \rightarrow I_y$. Total equivalences between x and y(i.e *teq(x,y)*), hold in cases where M is a total and onto isomorphism.

In the following, we present two belief measures on the non causality of attributes based on non common classes and refinements of attribute pairs.

Belief due to Non Common Classes. Belief on the domain equality condition of dependency mappings can be obtained by checking the existence of non common classes in the scopes of the involved attributes. Such classes indicate disjoint subsets in the domains of the attributes due to their different intensions.

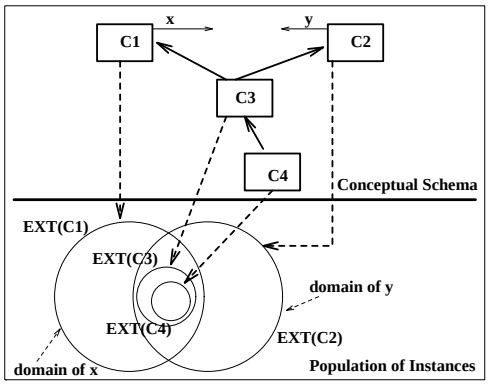

Figure 3: Non Common Classes of Attributes

Figure 3 illustrates this case. The differences in the properties comprising the intensions of the classes C1 and C2 are the logical conditions justifying the existence of elements in the sets (EXT[C1] - EXT[C2]) and (EXT[C2] - EXT[C1]).

Hence, we define a belief disfavoring the existence of any dependency mapping between two attributes i and j, as follows:

$$Bel(\overline{dep(i,j)}) = d_{ij} = \frac{(|S[i]-S[j]|+|S[j]-S[i]|)}{(|S[i]\bigcup S[j]|)}$$

Belief due to Non Common Refinements. Certain patterns of refinements of two attributes over the conceptual schema make impossible a total equivalence between them regardless of its ontology.

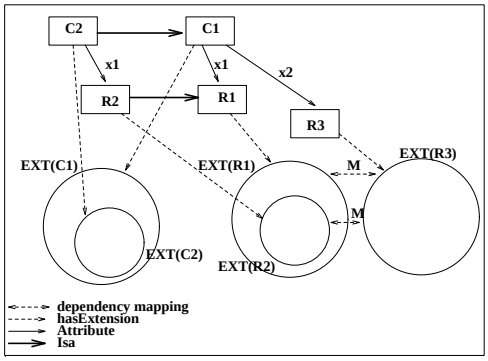

Figure 4: Non Common Refinements of Attributes

Figure 4 presents such a case. If an equivalence mapping between the attributes x1 and x2 was definable it would be necessary that their images with respect to objects in C1 and C2 would have identical cardinalities(i.e |EXT(R1)| = |EXT(R3)| and |EXT(R2)| = |EXT(R3)|). However, since x1 is refined in C2, it must be the case that |EXT(R2)| < |EXT(R1)|, which contradicts the previous constraint.

Hence, the belief to the non existence of a total equivalence mapping between two attributes i and j, due to their non common refinements over the schema is defined as:

$$Bel(\overline{teq(i,j)}) = r_{ij} = \frac{|(S[i]-AR[i])\bigcap AR[j]|+|(S[j]-AR[j])\bigcap AR[i]|}{|AR[i]\bigcup AR[j]|}$$

This belief function measures the cases where attribute i is refined over the conceptual schema while attribute j is not and vice versa. Like the belief due to the non common classes, the absence of non common refinements is not considered as evidence favoring the definability of a total equivalence mapping between i and j.

Estimating the Boundaries of the Salience Range. On the basis of the presented partial beliefs the boundaries of the salience range are estimated by the following formulas(the detailed derivation of these formulas can be found in [22]):

$(F1)$: $Bel(DOM_i) = e_u c_i$

u is the introducing class of attribute i

$(F2)$: $P^*(DOM_i) = (1-(1-c_i e_u)\prod_{j=1}^{n} d_{ij})$

$(F3)$: $P^*(DOM_i) = 1-(1-c_i e_u)(\prod_{j=1}^{n}(1-(1-d_{ij})(1-r_{ij})))$

n is the number of the other attributes of the classes in the scope of i.

F2 and F3 apply when all possible dependencies or only total equivalences between attributes are taken into account, respectively.

5. A Prototype of the Similarity Model

The similarity model has been prototyped and integrated with the Semantic Index System[2], an outgrowth of the Software Information Base, developed in the ESPRIT project ITHACA[3].

This initial integration augmented the general object management, querying, browsing and hypertext capabilities of the Semantic Index System with three similarity-based functionalities, namely:

i. the detailed similarity analysis between two TELOS objects, resulting into a graph identifying their analogous elements(see figure 7)

ii. the sorting of the instances of a class according to their similarity to some other object(see figure 6) and,

iii. the assingment of prototypicalities to the instances of some class, defined as average pairwise similarities as in [28](see figure 5)

In the next section we present some of these functionalities, through an example of using a version of this prototype(called the *Software Requirements Analyzer*), developed within the ESPRIT project NATURE[8], for specifying requirements of information systems.

6. Example: Specifying Requirements By Similarity Based Reuse

In this section, we exemplify how similarity and prototypicality may be utilized in specifying requirements for new information systems by reusing specifications of existing systems. In such a task, similarity analysis may be used in a complementary fashion with other forms of reasoning(e.g inheritance) and searching(e.g browsing).

Assuming the existence of a Software Information Base(SIB) storing specifications for a *Municipal Library System*, a *Car Rental System* and a *Motorbike Rental System*, described as aggregations of transactions, entities and agents according to the *domain* and *system* world in [14], the goal is to specify similar artifacts for a *University Library System*. Proceeding towards this goal, we focus on the specification of some transaction for returning books from their borrowers back to the library, namely the *ReturnLibraryItem* transaction. Similarity analysis may help in different ways:

i. As the basis for prototypicality analysis, it may be used for assigning prototypical examples(i.e instances) of

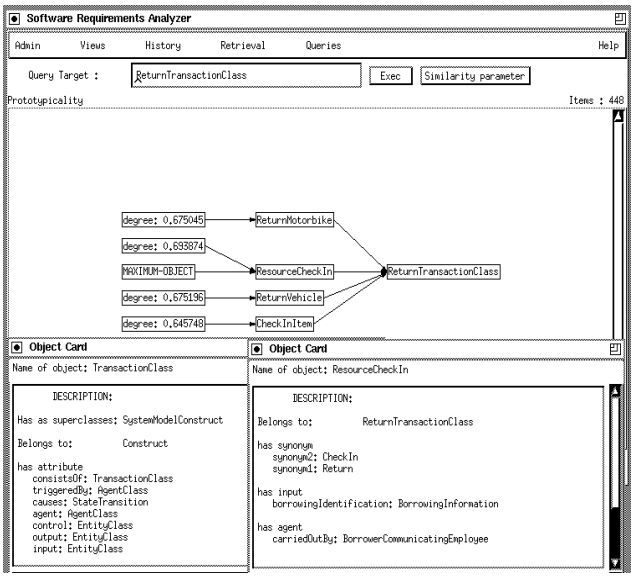

Figure 5: Selection of Prototypical Examples

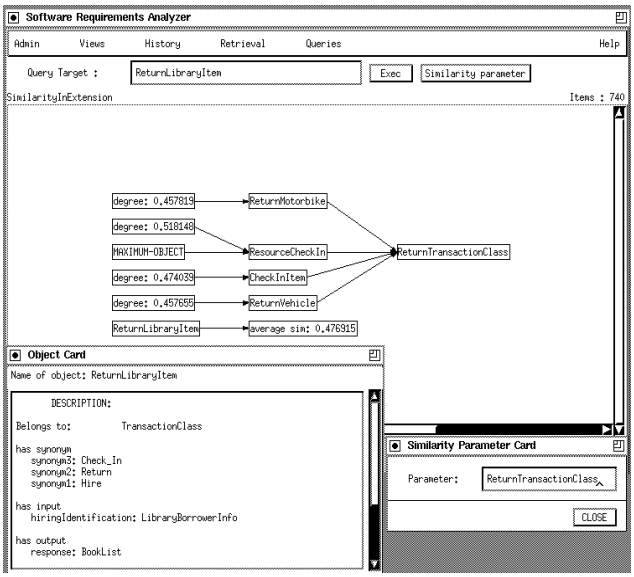

Figure 6: Ranked Retieval By Similarity

specification models indicating how such models could be used. Prototypicality is estimated as the average similarity of some specification with the other specifications built according to the relevant model. Figure 5, illustrates such a case. The most prototypical transaction *ResourceCheckIn* can be used as an example on how to specify transactions supporting the return of borrowed resources back to their holding systems.

ii. Similarity may also support ranked retrieval. As in figure 6, we can select and investigate the resource returning transactions in our repository, in descending similarity order with respect to a partial target specification of the *ReturnLibraryItem* transaction.

iii. The detailed similarity analysis between two specifications, resulting in a graph presenting their analogues and unique elements, may support the completion of a target specification in three distinct ways. The unique elements of the source specification may indicate an incompleteness in the target, and therefore they should be examined for reuse. For instance, there is no specification of who is responsible for the execution of the *ReturnLibraryItem* transaction, unlike the *CheckInItem* transaction(see *carriedOutby* element in figure 7), though it should.

On the other hand, both these transactions have elements identifying a past resource borrowing activity(i.e attributes *borrowingIdentification* and *hiringIdentification*) but in different ways. The *ReturnLibraryItem* exploits some general borrower information, unlike the *CheckInItem* transaction, which uses a unique code assigned to each copy of a book. Such analogous but slightly different ele-

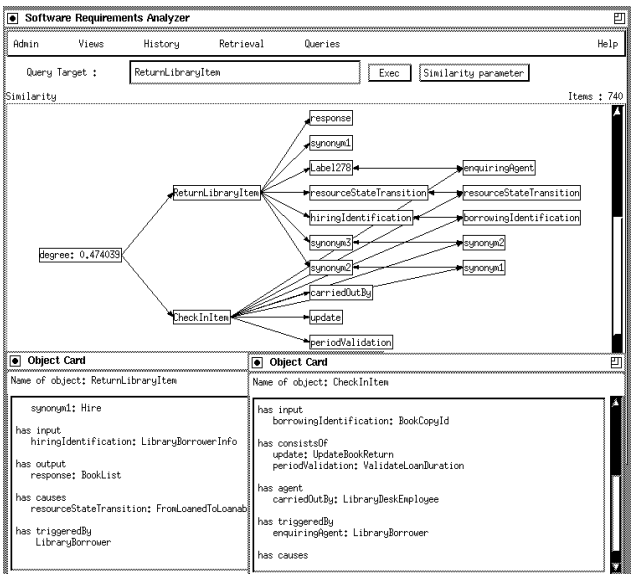

Figure 7: Similarity Between Transactions

ments may give rise to alternative solutions, as in this case where the code solution precludes any short of ambiguities in identification.

Notice also that elements which are unique to some target specification could be redundant and thus their presence should be further justified.

These examples illustrate how similarity analysis may support analogical reuse in different ways and can be used as a starting point for modeling the process of reuse by similarity analysis.

7. Conclusions

In this paper, we presented a general model for computing similarities between descriptions of software artifacts so as to promote their analogical reuse. Similarity estimates incorporate measures of salience of the attributes comprising the involved descriptions. The whole approach does not rely on any special form of knowledge but exploits only the semantics of general conceptual modeling abstractions in an attempt not to be knowledge acquisition intensive.

The role of similarity in reuse(partly demonstrated through an example of specifying requirements for information systems), is currently being elaborated through a process model. Furthermore, we intend to investigate the application of the model for integrating conceptual models of requirements specifications.

8. References

[1] D. Aha , Case-Based Learning Algorithms, Proceedings of the DARPA Case-Based Reasoning Workshop, 1991

[2] P. Constantopoulos , M. Doerr , The Semantic Index System: A Brief Presentation, ICS-FORTH, 1993

[3] P. Constantopoulos , M. Doerr , Y. Vassiliou , Repositories for Software Reuse: The Software Information Base, In Proceedings of the IFIP Conference in the Software Development Process, Como,Italy, 1993

[4] T. Cain , et al., Using Domain Knowledge to Influence Similarity Judgements, Proceedings of the DARPA Case-Based Reasoning Workshop, 1991

[5] R. Pietro-Diaz, Report: Software Reusability, IEEE Software, May 1993

[6] R. Pietro-Diaz , P. Freeman , Classifying Software for Reusability, IEEE Software, January 1987

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

[8] M. Jarke, et al., Theories Underlying Requirements Engineering: An Overview of Nature at Genesis, Proceedings of the 1st Symposium on Requirements Engineering, 1993

[9] R. Johnson, Designing Reusable Classes, Journal of Object Oriented Programming, June/July 1988

[10] M. Koubarakis, et al., TELOS: Features and Formalization, Technical Report KRR-TR-89-1, University of Toronto, 1989

[11] B. Meyer, Object Oriented Software Construction, C.A.R Hoare, Series Editor, 1989

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

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

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

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

[16] E. Ostertag, et al., Computing Similarity in a Reuse Library System: An AI-Based Approach, ACM Transactions on Software Engineering and Methodology, 1(3), July 1992

[17] W.B. Porter, et al., Concept Learning and Heuristic Classification in Weak-Theory Domains, Artificial Intelligence, 45, 1990

[18] E. Rosch, et all., Basic Objects in Natural Categories, Cognitive Psychology 8, 1976

[19] S. Russel, Analogy By Similarity, Analogical Reasoning, Kluwer Academic Publishers, 1988

[20] G. Spanoudakis, P. Constantopoulos, Similarity for Analogical Software Reuse: A Conceptual Modeling Approach, Proceedings of CAiSE '93, LNCS 685, June 1993

[21] G. Spanoudakis, P. Constantopoulos, Similarity for Analogical Software Reuse: A Computational Model, ICS-FORTH, Heraklion, Crete, September 1993 (submitted for publication)

[22] G. Spanoudakis, P. Constantopoulos, On Evidential Feature Salience ICS-FORTH, Heraklion, Crete, 1994 (submitted for publication)

[23] E. Sciore, Object Specialization, ACM Transactions on Office Information Systems, 7(2), 1989

[24] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, 1975

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

[26] K. Sycara, D. Navinhandria, Influences: A Thematic Abstraction for Creative Use of Multiple cases, DARPA Case-Based Reasoning Workshop, 1991

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

[28] A. Tversky, Features of Similarity, Psychological Review, 44(4), July 1977

[29] M. Weber, Implementation Models for the SIB, ITHACA.SNI.92.E2.#2, September 1992

[30] P. Wegner, Dimensions of Object Based Language Design, Proceedings of OOPSLA '87, 1987