GUIDELINES FOR EVALUATING CLASSES IN DATA MODELING

Jeffrey Parsons
The University of Manitoba

Yair Wand
The University of British Columbia

Follow this and additional works at: http://aisel.aisnet.org/icis1992

Recommended Citation
Parsons, Jeffrey and Wand, Yair, "GUIDELINES FOR EVALUATING CLASSES IN DATA MODELING" (1992). ICIS 1992 Proceedings. 47.
http://aisel.aisnet.org/icis1992/47

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 1992 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
GUIDELINES FOR EVALUATING CLASSES IN DATA MODELING

Jeffrey Parsons
Faculty of Management
The University of Manitoba

Yair Wand
Faculty of Commerce and Business Administration
The University of British Columbia

ABSTRACT

The notion of class is widely used in systems analysis and data modeling methodologies and is fundamental to emerging object-oriented approaches. However, there are no theory-based guidelines for evaluating the quality of a collection of classes selected to model an application. In this paper, we assume that the class is used as a modeling construct in an effort to follow more closely the way humans structure knowledge about things in the world. We propose necessary conditions for a set of classes to be considered good. These criteria take the form of four principles of classification and are derived from research in cognitive psychology and linguistics about the importance of categories and concepts. That research suggests that humans form categories in order to provide cognitive economy and to support inferences about classified instances in the absence of complete information. We define the notions of potential class and class structure, which satisfy the principles of classification, and discuss some implications of failing to follow these principles when deciding on the classes needed in an application. Additionally, we show that multiple class structures over a given domain are supported.

1. INTRODUCTION

Use of the class as a modeling construct in various aspects of information systems development is becoming increasingly widespread. Classification was first proposed as a useful abstraction mechanism in logical data modeling (Chen 1976; Smith and Smith 1977; Teorey, Yang and Fry 1986), and later adopted by various semantic data models (Hammer and McLeod 1981; Shipman 1981; Brodie 1984; Hull and King 1987; Peckham and Maryanski 1988) and object-oriented data models (Banerjee et al. 1987; Fishman et al. 1987; Hornick and Zdonik 1987; Bries et al. 1989). From somewhat different origins, classes have played an important role in object-oriented programming languages (Robson 1981; Rentsch 1982; Pascoe 1986; Nierstras 1987; Stroustrup 1987; Goldberg and Robson 1989) and in knowledge representation languages (Bobrow and Winograd 1977; Brachman and Schmolze 1985; Fikes and Kehler 1985; Stek and Bobrow 1986). Most recently, classification has been "re-introduced" to systems development as an important part of several object-oriented systems analysis and design methodologies (Bailin 1989; Henderson-Sellers and Edwards 1990; Wirfs-Brock and Johnson 1990; Booch 1991; Coad and Yourdon 1991).

A class serves the role of abstracting or summarizing what is common about a set of instances (entities, objects, things). Informally, a class is a collection of attributes (and possibly behavior) shared by a set of instances. Classes may be related to other classes through a hierarchy or lattice, such that more specialized classes inherit attributes from more general ones.

Various reasons have been offered for the importance of classes in each of the areas above. Some advocate implementation advantages — such as their value in creating instance objects which possess desired variables, in hiding details about the implementation of methods, and as repositories for methods (Ledbetter and Cox 1985; Stek and Bobrow 1986; Banerjee et al. 1987). Other reasons suggest representation advantages — such as the value of classes in reflecting the way humans organize knowledge about things (Banerjee et al. 1987; Digitalk 1989; Goldberg and Robson 1989; Mylopoulos 1990; Coad and Yourdon 1991; Parsons and Wand 1991; Mylopoulos and Motschnig-Pitrik 1992).

In light of the growing importance of classes in modeling a domain during systems analysis, it would be valuable to have guidelines by which an analyst might evaluate whether
a given choice of classes is "good." However, the literature on classification in data models does not provide a foundation for such an evaluation. Although mechanisms are sometimes offered for constructing classes (e.g., Hammer and McLeod 1981; Goldberg and Robson 1989; Coad and Yourdon 1991), these do not help in determining whether a selected set of classes is an appropriate model. We believe that this lack of guidelines impairs the usefulness of classification as an abstraction mechanism. Based on the assumption that data modeling involves creating a representation of users' knowledge about some subject matter, we propose specific guidelines for evaluating the quality of a set of classes as a representation of pertinent knowledge. These guidelines are derived from two basic assumptions from cognitive psychology about the role of classification in human survival.

2. COGNITIVE FOUNDATION

A great deal of work has been done in cognitive psychology and linguistics (among other of the cognitive sciences) which studies the importance of classification to the way humans organize knowledge about the world. The formation of concepts to describe what is common about individuals and the classification of further individuals into these categories is essential to human survival and adaptation. For example, Lakoff (1987) states:

"Without the ability to categorize, we could not function at all, either in the physical world or in our social and intellectual lives. An understanding of how we categorize is central to any understanding of how we think and how we function (p. 6)."

Systems analysis has been claimed to be essentially a modeling activity (Olle et al. 1988). Accordingly, the role of constructs used in this activity is to capture human knowledge about entities relevant to an application. Since concepts or categories are a vital mechanism for organizing human knowledge, understanding why and how they are used may be crucial to the proper use of classification abstractions in systems analysis and data modeling.

There are two primary functions of concepts (Smith 1988). The first is to support what has been called cognitive economy (Rosch 1978). Grouping many instances into a single category identifies them as being the same in certain respects. Usually, this can be taken to mean that the instances share some set of properties and this shared set is referred to as a concept. Properties may express facts about states, relationships among individuals, or constraints on allowed states and behavior (Parsons 1992). Concepts permit reasonable questions to be asked with respect to known instances of the concept. For example, if EMPLOYEE has as one its defining properties Department, meaning that each employee works in one department, then it makes sense to ask "which department does x work in?" for any x who has been previously classified as an employee. Similarly, given a new individual y, if we are told that y works in a department, we may (if no other known concepts have this property) be able to classify y as an EMPLOYEE.

The second important role of concepts emerges from the previous statement. If an individual is classified in a particular way based on only some of its properties, one can infer additional properties of interest that the individual must possess. That is, possessing one subset of properties may imply that an individual possesses another subset. For example, people learn from experiences with fires that they produce flames and generate heat. Consequently, on observing a distant fire by the presence of flames, an individual "knows" that it will be hot, even if the heat is not felt at that distance.

A theory of concepts generally specifies several things. First, it indicates what a concept is (e.g., an abstraction consisting of a set of properties) (Medin and Smith 1984). Second, it provides mechanisms for determining whether an instance belongs to a category (Bruner, Goodnow and Austin 1956). Third, a comprehensive theory should describe how concepts are related to other concepts in memory (Smith 1978). Finally, a theory of concepts should offer an explanation of how concepts are acquired.

There are several theories of the nature of concepts and categorization. The classical view holds that a concept is defined in terms of a number of individually necessary and jointly sufficient conditions for membership of a thing in a category. The prototype view treats a concept as a probabilistic representation which expresses the most typical member of a category. Unlike the classical view, it does not require that all instances possess a defining set of properties. Both these theories view concepts as intensions (i.e., abstractions defined by absolute or probabilistic conditions of membership). A third theory is based on exemplars, and regards a concept as an extension only (i.e., the definition is by enumerating the instances that are members). Instances are grouped together by similarity on one or more dimensions and different instances may be similar in different ways.

In this paper, we use four formal constructs to describe knowledge about an application domain (Parsons 1992):

1. An instance is viewed as a symbol uniquely designating the existence of a thing in the world;
2. A property is a function from a set of instances to a set of values, where the values may be other instances (e.g., the supplier that supplies a part);
3. A concept is a set of properties;
4. One concept is a specialization of a second if and only if the set of properties of the first is a strict superset of those of the second.

The literature on concepts and classification devotes considerable attention to concept acquisition (Bruner, Goodnow
and Austin 1956; Rosch 1973, 1978; Medin and Smith 1984; Glass and Holyoak 1986; Lakoff 1987). In what follows, we assume that concepts are acquired through a series of steps. Humans are born and perceive individual things in the world. Over time, certain properties are identified as being shared among things (e.g., height, weight). In addition, certain groups of properties are aggregated to form concepts (e.g., person). We further assume that concept acquisition proceeds in a manner that satisfies several principles which are based on the importance of cognitive economy and inference as fundamental reasons for dividing knowledge of the world into categories. We propose that these principles should guide the selection of classes to model a domain, formalize what is required for the principles to be satisfied, and demonstrate their use by examples.

3. POTENTIAL CLASSES

Systems analysis and data modeling approaches generally assume that the set of classes to be modeled is known in advance by the prospective users of the system under consideration. The emphasis of these efforts is often on an intensional view of classes, since they focus on defining the attributes which make up classes. While we concur that a class is properly an intension, the principles which follow are developed by explicitly considering that classes are formed in order to abstract the similarity of instances. Instances are viewed as preceding concepts and their role in determining the quality of classes is significant. In other words, we are interested not just in what the classes are, but also in why a particular set of classes is chosen. In this section and the next, we propose that several principles govern concept formation and use these as guidelines for restricting which sets of properties can constitute a (potential) class and which groups of (potential) classes can coexist.

**Principle of Abstraction from Instances:**

A concept abstracts the properties shared by a non-empty set of instances.

In perceiving the world, we classify instances according to the properties they share with others. If a concept is defined as a set of properties, there must be a non-empty set of instances (corresponding to real or imagined things) which possess all of these properties. Otherwise, the concept is not useful. This supports cognitive economy, since a concept which has no perceived extension abstracts nothing about known entities. In addition, we propose a second principle to limit which collections of properties may constitute a class.

**Principle of Maximal Abstraction:**

Every property possessed by all the instances of a concept is part of the definition of that concept.

This principle also supports cognitive economy. According to Rosch, "the task of category systems is to provide maximum information with the least cognitive effort" (1978, p. 28). In other words, classifying an instance as belonging to a concept enables certain questions to be answered about that individual. Clearly, if a concept definition omits some properties which are possessed by all its instances, it would not be possible to answer questions related to that property when an instance is classified. For example, there should be no concept defined only by the property Work_in_department, since all who work in departments are persons and, therefore, share numerous other properties. In other words, if one can ask of an instance x, "Which department does x work in?", one can also ask "What is x's address?", "What is x's birthdate?", and so on.

One might argue that, for a given application, the property Work_in_department is sufficient to identify instances of a class EMPLOYEE and therefore may be the only property defining the class. However, in common knowledge, employees are persons. Therefore, if there are other properties in the domain of interest whose domain is a superset of that of Work_in_department (e.g., Address, Birthdate), these must also be properties of EMPLOYEE. It is useful to refer to properties which are sufficient to classify instances, but which do not constitute all of properties of the class, as identifying properties.

These two principles allow the definition of potential class. Before doing this, we recognize the connection between a class as a set of properties (an intension), and its extension, which is a set of instances. The domain of a property P, denoted dom(P), is the set of things (instances) that possess this property. The extension of any set of properties P, denoted ext(P), is the (joint) intersection of the domains of these properties.

In the following definition, T denotes the complete set of objects of interest (universe) in an application domain, P = \{P₁,...,Pₜ\} denotes the set of all properties possessed by any members of T.

**Definition:**

C ⊆ P will be called a potential class in T iff

a) ext(C) ≠ ϕ, and

b) \( \exists P ∈ (Pᵢ−C) \) such that ext(C) ⊆ dom(P).

This definition ensures that every potential class (a) has a non-empty extension and (b) contains all properties possessed by all instances in its extension. Hence, it satisfies the principles of abstraction from instances and maximal abstraction. The example given earlier, where C = \{Work_in_department\}, does not constitute a potential class, since all instances which possess the property Work_in_department also possess numerous other properties, such as Address and Birthdate. Hence, the domain of the latter two properties is a superset of that of the first.
The definition of potential class leads to a theorem about an important characteristic of distinct potential classes which has implications for class (and concept) specialization. Note, first, that two distinct potential classes cannot have the same set of properties (else they are the same).

**Theorem:**

Two potential classes cannot have the same extension.

**Proof:**

Let the classes be A and B and assume they have the same extension. If the classes are distinct, then at least one possesses a property that the other does not. Assume it is A and the property is \( P' \). This means that the extension of class B also possesses \( P' \), which contradicts the definition of potential class.

**Implications:**

To demonstrate the value of potential class, consider the following example. An information system is being developed for a member-owned cooperative whose customers are exactly its owners. Suppose that one user, interested in distributing the product catalogue, identifies two classes of individuals who would be interested in the catalogue: CUSTOMERs and OWNERs. If the properties of interest to this user have the same domain, there is no point in distinguishing two classes. Having more than one might potentially result in sending the catalogue twice to all individuals. Some readers may find difficulty with this view, since everyone "knows" that CUSTOMER and OWNER are different concepts. We concur, but suggest that the reasons for this difference must be explicitly stated.

In a larger scope (e.g., our knowledge of all companies), not all customers are owners and vice versa. Hence, in an analysis situation, users bring this "conceptual baggage" to the table. By contrast, if the user in question has no other experience, only one class is possible if maximal abstraction is to be satisfied. What makes the example interesting, however, is that another user in the same organization, who is interested in a different application, may view the sets of properties of interest as having different domains (or even use the same labels for different sets of properties). For example, in an accounting application, OWNERs appear in payables (together with SUPPLIERs) and CUSTOMERs in receivables. From this user’s point of view, the classes CUSTOMER and OWNER may possess different properties and need to be distinguished.

Note that the analysis in this example requires that the extension of the classes be known. Consequently, we predict that an analysis technique which identifies classes only, without explicitly considering instances, will be of little use in identifying this kind of situation.

In the next section, we introduce two further principles which should guide the selection of classes for an application.

4. **CLASS STRUCTURES**

This section develops two principles which restrict the nature of the potential classes that can co-exist if cognitive economy and inference are to be supported and defines a construct — the *class structure* — to characterize a set of such classes. The first principle, if followed, guarantees that all relevant knowledge is used to form classes.

**Principle of Completeness:**

Given a "universe of knowledge" consisting of (1) a set of instances, (2) the set of all properties possessed by any of these instances, and (3) a set of concepts, every property will be used in the definition of at least one concept in that set.

This principle essentially states that no properties are omitted in the abstraction of some universe of knowledge. If a property was not abstracted in the definition of any concept, there would be some questions about the set of instances possessing that property which could not be answered by classifying those instances into any available concept. In other words, properties are a crucial classification tool and are not normally considered independently of the concepts they define.

The second principle eliminates classes that can be inferred from other classes.

**Principle of Non-redundancy:**

A concept whose properties constitute a superset of those of each of several other concepts must contain at least one property not in the union of the properties of the other concepts.

This means that a concept that "specializes" one or more other concepts must contain information (i.e., additional properties) which cannot be found by considering any instance as a member of any of the more general concepts. This principle is based on a premise, implicit in many theories of semantic memory (e.g., Smith 1978), that conceptual organization is efficient in the sense that each refined concept adds new knowledge.

Given a "universe" of knowledge (a domain), there may be many ways of forming concepts about that domain which satisfy the four principles introduced above. As a consequence, there is no single "correct" set of concepts for abstracting one's knowledge about an application (cf. Lakoff 1987), as suggested by the example of the customer-owned cooperative. Instead, any "world view" which satisfies the principles is as valid as any other. Different individuals may abstract different sets of properties to form different concepts depending on various factors, such as the context within which the knowledge will be used (e.g., Barsalou 1982; Rosch 1978) and the culture within which the individual lives (Lakoff 1987). However, any abstraction which violates any of the principles violates some elementary assumption about the nature of classification.
A set of classes used to organize knowledge about a domain will be called a class structure. A class structure must satisfy the principles of non-redundancy and completeness, thereby imposing some constraints on the nature of the potential classes it contains. In addition, there may be many possible class structures which model a given subject matter.

Let \( T \) denote a “universe” of instances, \( P^T = \{p_1, ..., p_m\} \) denote the set of all properties possessed by any members of \( T \), and \( CS = \{C_1, ..., C_k\} \) denote a set of potential classes.

**Definition:**

\( CS \) is a class structure over \( T \) iff

1. \( \cup_{i \in \{1, ..., k\}} C_i = P^T \), and
2. \( \forall C \in CS \) such that \( C \) is the union of any subset of \( (CS-C') \).

If \( CS \) is a class structure, each potential class \( C \in CS \) will be referred to simply as a class. By its definition, a class structure (a) contains every property possessed by any members of \( T \) in at least one of its classes and (b) contains no class whose properties are exactly the union of the properties of any other classes in the structure. The second condition is important in defining class specialization. It prohibits a class structure from containing potential classes which do not have at least one extra property with respect to its superclass (if only one), as well as potential classes which simply contain the union of the properties of two (or more) superclasses. This is a significant departure from existing approaches to supporting subclasses (cf. Hammer and McLeod 1981; Coad and Yourdon 1991). For example, SDM (Hammer and McLeod 1981) allows subclasses to be defined based on values of properties, intersections of the extensions of other classes, and other criteria.

To illustrate the definition, an example of a collection of potential classes which are not part of the same class structure is given. Suppose that \( CUSTOMER = \{Name, Address, Spouse, Holds_account\} \), \( EMPLOYEE = \{Name, Address, Spouse, Experience\} \), and \( CUST_EMP = \{Name, Address, Spouse, Holds_account, Experience\} \). In this case, \( \{CUSTOMER, EMPLOYEE, CUST_EMP\} \) does not constitute (a subset of) a class structure since \( CUST_EMP \neq CUSTOMER \cup EMPLOYEE \). This means that, as defined, the potential class \( CUST_EMP \) abstracts no additional information with respect to the other concepts considered, since all questions that can be answered with respect to any instance of \( CUST_EMP \) can be answered with respect to that instance as a member of \( CUSTOMER \) or \( EMPLOYEE \).

The following theorem shows that multiple class structures over a given domain are possible.

**Theorem:**

A domain of more than one property can be described by more than one class structure.

**Proof:** (by example)

Let \( P^T = \{p_1, p_2\} \) denote a universe of properties, \( C^1 = \{p_1\} \), \( C^2 = \{p_2\} \), and \( C^3 = \{p_1, p_2\} \) denote potential classes. \( C^1, C^2 \) is one class structure and \( C^3 \) is another.

The class is the fundamental abstraction mechanism for comparing instances. In addition, classes can be associated with other classes based on the degree to which they share properties. This leads to the consideration of class specialization. There is a great deal of evidence suggesting that we organize knowledge about things into general and more specialized categories. Biological taxonomies are an example (Keil 1979).

Given the definitions of potential class and class structure, the notion of class specialization can be formalized. Intensionally, specialization means that the set of properties possessed by a specialized class is a strict superset of the set of properties of a more general class. Specialization also has the extensional meaning that the instances of a specialized class are a strict subset of those of a more general class and, therefore, are also instances of the latter. The definition will be shown to satisfy this condition.

**Definition:**

Let \( C^1 \) and \( C^2 \) denote two classes in a class structure. Then, \( C^2 \) is a specialization (or subclass) of \( C^1 \) (alternatively, \( C^2 \) IS-A \( C^1 \)) iff \( C^1 \subseteq C^2 \).

For example, if \( PERSON = \{Name, Address, Birthdate\} \) and \( CUSTOMER = \{Name, Address, Birthdate, Holds_account\} \) are two classes in a class structure, then \( CUSTOMER \ IS-A \ PERSON \). If \( C^2 \) is a subclass of \( C^1 \), \( C^1 \) will be referred to as a superclass of \( C^2 \).

If a class is a specialization of two or more others, it must possess added properties with respect to the union of those of all its superclasses. Otherwise, all questions about instances can be answered with respect to one or more of the more general classes. This is implied by the definitions of class structure and specialization, as shown by the following theorem.

**Theorem:**

Let \( C^1, C^2 \) denote two classes in a class structure such that neither is a specialization of the other. Then, if \( C^1 \) is a specialization of both \( C^2 \) and \( C^3 \), \( C^1 \cup C^2 \subseteq C^3 \).

**Proof:**

\( C^1 \) is a strict superset of each of \( C^2 \) and \( C^3 \), since it is a specialization of each. But since all three are part of the same class structure, \( C^2 \) is not the union of the properties of \( C^3 \) and \( C^1 \). Therefore, it is a strict superset of the union of the two.
Implications:

Consider the cooperative discussed earlier. Suppose that the class **OWNER_EMPLOYEE** is suggested as a subclass of both **OWNER** and **EMPLOYEE** so that the fact that some owners are employees is explicitly recognized. If, however, this is the only reason for making the distinction and there are no additional properties of the proposed class (other than those of both **OWNER** and **EMPLOYEE**), the principle of non-redundancy is violated. **OWNER_EMPLOYEE** is redundant in the sense that all questions that can be answered (or properties that can be inferred) by knowing that an instance belongs to **OWNER_EMPLOYEE** can also be answered (inferred) by knowing that the instance belongs to both **OWNER** and **EMPLOYEE**. Hence, the subclass would not, under these circumstances, be considered valid. It is more likely that further investigation by the analyst and users would lead to the conclusion that there are added properties associated with the instances of **OWNER_EMPLOYEE** that do not apply to **OWNERS** or **EMPLOYEES** in general. For example, these individuals may be able to take part of their salary in shares of the cooperative instead of in cash. In that case, the subclass may well be justified for the application.

The definition of specialization satisfies the basic extensional intuition—that the extension of a specialized class is a strict subset of that of a more general one.

**Theorem**

Let \( \text{ext}(C^1) \), \( \text{ext}(C^2) \) denote the extensions of the potential classes \( C^1 \) and \( C^2 \), respectively, where \( C^2 \) is a subset of \( C^1 \). Then, \( \text{ext}(C^2) \subseteq \text{ext}(C^1) \).

**Proof:**

\( C^1 \) contains a strict subset of the properties of \( C^2 \) since \( C^2 \) is a subset of \( C^1 \). Additionally, there is no property which is part of the definition of \( C^2 \) but not of \( C^1 \), whose domain is a superset of \( \text{ext}(C^1) \). Therefore, the shared domain of the properties of \( C^2 \) which are not part of \( C^1 \) is not a superset of \( \text{ext}(C^1) \). But the extension of \( C^2 \) is the intersection of \( \text{ext}(C^1) \) with the common domain of the additional properties of \( C^2 \). Therefore, \( \text{ext}(C^2) \) is a strict subset of \( \text{ext}(C^1) \).

**Implications:**

The principle of non-redundancy implies that a new subclass should possess additional properties with respect to its superclasses. As a consequence, this prevents the definition of subclasses based only on differentiating instances of an existing class by specific values of one or more properties. Consider again the customer-owned cooperative. An example of an attempt to define subclasses based on values might be the distinction of two "specializations" of **EMPLOYEE**—those who work at the **Vancouver** store versus those who work at the **Seattle** store. Such subclasses are useful, in the sense of cognitive economy and inference, only if they also possess additional properties.

Of course, values of a property may serve as a convenient mechanism for distinguishing instances of classes. It may be that employees in the **Seattle** office pay premiums to receive a company health plan. **Vancouver** employees need not possess this property since they benefit from universal government-funded basic medical care. However, if there are no added properties, we predict that an artificial distinction is not useful.

5. CONCLUSIONS

The explicit use of classification mechanisms in modeling the entities of interest in information systems applications is growing in importance, especially in light of the recent adoption of object-oriented techniques in systems analysis and design. Selecting appropriate classes to model a domain is an important aspect of systems analysis. Despite this, many approaches to systems development provide mechanisms for defining classes in terms of common attributes without providing guidelines for evaluating classes.

This work is based on the assumption that classification mechanisms are used to create categories of instances that correspond to the concepts with which humans model knowledge about things in the world. Using foundations from cognitive psychology and linguistics about the basic reasons for classifying things, we propose four principles which should guide the choice of classes for a particular application. A choice which does not satisfy these principles is shown to contain some deficiency. The principles provide the basis to formalize two notions—potential class and class structure—which together provide necessary conditions for a set of classes to be considered good. The result is a prescription for systems analysts: namely, that a **proposed collection of classes should constitute a class structure**. An interesting consequence of this prescription is that it supports multiple views of an application by recognizing that different class structures can be used to model precisely the same domain of instances and properties.

The field of classification theory may offer many additional insights for systems analysis and data modeling. Of special interest is the question of limiting possible class structures by identifying sufficiency criteria (according to some measure of "goodness"). Another issue we are currently examining is that of providing support for flexible class structures to reflect multiple users interests. We hope that further work will generate more guidelines for conducting systems analysis, as well as a blueprint for implementing a system which supports dynamic views of classes in an application.
6. ACKNOWLEDGEMENTS

This research was supported in part by research grants from the Faculty of Management at The University of Manitoba to Jeffrey Parsons and from the Natural Sciences and Engineering Research Council of Canada to Yair Wand.

7. REFERENCES


Hornick, M., and Zdonik, S. “A Shared, Segmented Memory System for an Object-oriented Database,” ACM Transactions on Office Information Systems, Volume 5, Number 1, January 1987, pp. 70-95.


9. ENDNOTES

1. We use the term "instances" to refer generically to any of these.

2. Note that this is an intensional view, which has an associated extension. A precise definition of class (in terms of attributes) is given in section 3.

3. A detailed explanation of these theories, as well as a review of the empirical evidence pertaining to each, is given in (Smith and Medin 1981).

4. Since values may be instances, relationships among instances are modeled as properties. Treatment of behavioral constraints as properties is not considered here, but can be handled somewhat differently (e.g., Parsons 1992).

5. We do not address here the reasons for particular aggregations other than that collections of properties are chosen because they are useful in some context (cf. Lakoff 1987).

6. Note that if the intension of class C (i.e., the set of defining properties) is a superset of the intension of class C, then the extension of class C (the set of actual members) is a subset of the extension of class C.

7. VANCOUVER_EMPLOYEES should not form a separate class unless it possesses properties not possessed by SEATTLE_EMPLOYEES.

8. We do not claim that these are sufficient conditions for goodness. More work is required to determine if additional useful conditions can be formalized.