MIS Quarterly

SPECIAL ISSUE

USING COGNITIVE PRINCIPLES TO GUIDE CLASSIFICATION IN INFORMATION SYSTEMS MODELING[1]

By: **Jeffrey Parsons**
Faculty of Business Administration
Memorial University of Newfoundland
St. John's, NL A1B 3X5
CANADA
jeffreyp@mun.ca

Yair Wand
Sauder School of Business
The University of British Columbia
Vancouver, BC V6T 1Z2
CANADA
yair.wand@ubc.ca

Abstract

Organizing phenomena into classes is a pervasive human activity. The ability to classify phenomena encountered in daily life in useful ways is essential to human survival and adaptation. Not surprisingly, then, classification-oriented activities are widespread in the information systems field. Classes or entity types play a central role in conceptual modeling for information systems requirements analysis, as well as in the design of databases and object-oriented software. Furthermore, classification is the primary task in applications such as data mining and the development of domain ontologies to support information sharing in semantic web applications. However, despite the pervasiveness of classification, little research has proposed well-grounded guidelines for identifying, evaluating, and choosing classes when modeling a domain or designing information systems artifacts. In this paper, we adopt the cognitive notions of inference and economy to derive a set of principles to guide effective and efficient classification. We present a model for characterizing what may be considered useful classes in a given context based on the inferences that can be drawn from membership in a class. This foundation is then used to suggest practical design rules for evaluating and refining potential classes. We illustrate the use of the rules by showing that applying them to a previously published example yields meaningful changes. We then present an evaluation by a panel of experts who compared the published and revised models. The evaluation shows that following the rules leads to semantically clearer models that are preferred by experts. The paper concludes by outlining possible future research directions.

Keywords: Conceptual modeling, classification principles, classes and types, information modeling, design science

Introduction

Classification (or categorization) is a fundamental human capability (Lakoff 1987, p. 6). Humans immediately classify phenomena they encounter or, failing that, wonder what they "are." Moreover, we communicate about phenomena by referring to the class(es) to which we have assigned them. Current thought on classification holds that classes do not exist independently, but are constructed as *useful abstractions* of the similarities of the classified phenomena. Classification serves several purposes. First, it supports *inferences* about unobserved properties of an object derived by classifying it (Rosch 1978; Smith and Medin 1981). Classification also enables inference about other (concurrent) phenomena, and

[1]Ramesh Venkataraman was the guest associate editor for this paper.

about possible future states of us and our environment, and thus might affect our actions. To the extent that such inferences are valid, they enable humans to survive and adapt (e.g., by taking appropriate actions, such as finding food or avoiding danger). To the extent that inferences fail, they inhibit adaptation and survival. Simply put: "Categorization is about keeping the creature alive" (Winter 1998, p. 1). Second, classification reduces the complexity of dealing with instances. Instead of having to remember every feature of every instance, we classify instances and obtain knowledge of features of the class without having to associate them with every instance encountered (Rosch 1978; Smith and Medin 1981). Third, classification supports reasoning by assigning properties to the class as an abstraction and reasoning about them. For example, one might determine the average annual sales of the class "salesperson" and look for trends and causes of variations.

Information systems represent sets of concepts humans use to organize knowledge about domains. Concepts are manifested as *classes* or *types* in information systems development, and appear in artifacts such as databases, software, and formal ontologies. According to Sowa (2000, p. 51), "the selection of categories determines everything that is represented in a computer application." Therefore, the principles that humans use to classify phenomena are relevant to the development of IS artifacts. In particular, identifying domain concepts, manifested as classes, is an important part of conceptual modeling and early requirements identification. Thus, improving approaches to identifying classes can help improve requirements elicitation and representation. Properly identifying and stating requirements is critically important to the success of IS projects, and, hence, of vital concern to both senior managers and managers of IS projects (Hofmann and Lehner 2001; Wand and Weber 2002).

Despite the importance of classification, no well-grounded guidelines exist for choosing classes in information systems modeling. Consequently, different analysts can produce very different models of a given domain. We address this problem using design science guidelines (Hevner et al. 2004). We propose a design artifact consisting of a *model* of good classification structures and a *method* (rules) for constructing such structures. We apply the knowledge base of cognitive science classification principles (i.e., people organize phenomena in a domain into specific classes in order to process and retain information effectively) to develop and formalize a model and rules for constructing good classes. The model and rules are the first to take this approach to guide classification in the context of information systems modeling. We illustrate the practical usefulness of the artifact to information systems analysis by applying the method to a published conceptual schema, identifying key modifications as a result. We empirically evaluate the effectiveness of the method by having a panel of experts assess the two schemas. We focus explicitly on the experts' reasoning about their preferences between the original and revised schemas, and on relating that reasoning to the proposed modeling rules. Our findings indicate that the rules produced schemas that appeared to the experts to have clearer semantics and, hence, were preferred by them, thus providing evidence for the usefulness of the rules.

Classification in the Design of Information Systems Artifacts

Choosing and defining classes is a critical activity in many areas of information systems development. We briefly examine how classification issues arise in three representative areas: data management, object-oriented systems design, and ontological engineering. We first explore how classification is used in each of the three areas, and then examine research on how to choose classes in the context of data modeling and object-oriented analysis and design.

Classification Uses in IS Development

Data Management

Classes constitute a primary mechanism for imposing a structure on the data requirements for an information system. Other modeling constructs, including attributes, relationships, (class) hierarchies, and part–whole structures, are defined with respect to classes. Classification issues arose in early hierarchical and network data models (Tsichritzis and Lochovsky 1976), as well as in the set-based fixed-table structures of the relational model (Codd 1970). More complex issues in classification dominated research on conceptual and semantic data models (e.g., Chen 1976; Hammer and MacLeod 1981; Hull and King 1987; Kent 1979; Schmid and Swenson 1975; Smith and Smith 1977; Teorey et al. 1986).

Semantic modeling methods clearly recognize the importance of classes in describing the structure of knowledge about things in a domain. All methods provide constructs for representing classes, typically as a set of attributes/properties, and relationships/associations to other classes. In addition, semantic data modeling assumes that users and analysts interact to determine the classes, frequently with the analyst probing to uncover additional information or implicit assumptions, and the designer using the modeling language to represent these classes.

Object-Oriented Systems Design

Classification in object-oriented systems development is closely related to its role in data modeling. However, object-oriented programming languages introduce the additional concept of programming with objects that ***encapsulate*** structure (data) and behavior (methods). With few exceptions (e.g., Lieberman 1986), objects are created as instances of classes. Hence, class identification is a critically important activity in object-oriented analysis and design (Monarchi and Puhr 1992; Parsons and Wand 1997b). In particular, the unified modeling language (UML) deals extensively with the modeling of classes via class diagrams, which are the most widely used diagrams in the language (Dobing and Parsons 2006).

Ontological Engineering

With the emerging concept of the semantic web (Berners-Lee et al. 2001) and the need to standardize terminology and concepts for information sharing across organizations and applications, interest has grown in developing formal ontologies for a wide variety of domains, ranging from business to medicine to science (Fensel 2003).

An ontology is a specification of "classes of objects and relations that exist in some domain" (Chandrasekaran et al. 1999, p. 21). In that context, effective ontological engineering depends on defining a "good" set of classes to describe the domain. However, the literature on ontology building tends to focus on techniques and tools for specifying ontologies (e.g., Gruber 1993). Such mechanisms are usually silent about how to choose the classes of interest.

In sum, classification is pervasive in areas of the information systems field that involve the design and construction of artifacts to represent a domain. An appropriate choice of classes can have a significant effect on the quality of the designed artifacts such as databases and software. Next, we consider how the challenge of identifying and evaluating classes has been addressed in prior work on data modeling and object-oriented analysis and design.

Approaches to Class Identification and Evaluation

A range of practitioner and academic literature on entity–relationship modeling and class modeling proposes approaches to identifying and evaluating classes. Class quality has also been examined in the area of object-oriented software design (e.g., Bansiya and Davis 2002). However, we do not examine that domain since the guidelines proposed in such literature are implementation-oriented, focusing on issues such as cohesion, coupling, and optimizing the assignment of methods to classes to speed program execution (Dean et al. 1995).

Numerous authors have noted that classification is the central problem in object-oriented analysis and acknowledge that no well-justified, agreed-on principles exist. For instance, Overmyer et al. (2001, p. 401) state that

> despite the advantages that object technology can provide to the software development community and its customers, the fundamental problems associated with identifying objects, their attributes, and methods remain. These tasks are largely manual processes driven by heuristics that analysts acquire through experience.

Likewise, Song et al. (2004, p. 216) claim that

> finding a set of domain classes is the most important skill in developing an object-oriented system. However, finding classes is a discovery process. Discovering a set of domain classes is intellectually challenging and time-consuming.

Broadly speaking, existing literature focuses on three questions: (1) What constitutes a class? (2) How do we identify classes? (3) When should subclasses of a class be defined?

What Constitutes a Class?

There is general agreement that a class is an abstract description of a set of properties shared by a set of instances. In a database context, the properties are manifested as attributes and relationships (Elmasi and Navathe 2006). In an object-oriented context, the properties include attributes and operations/services, the latter dealing with object behavior (Booch et al. 1999). From this, we can conclude that any type of phenomenon that cannot be described in terms of attributes (and behavior) is not a candidate class. In conceptual modeling, an additional condition has been proposed: a class abstracts all of the relevant properties shared by its instances (Parsons and Wand 1997a). Beyond this, in the ontological engineering domain, the Ontoclean project (Guarino and Welty 2002) defines notions of *essence*, *rigidity*, *identity*, and *unity* as principles for classification in a taxonomy. However, in our view the proposed principles are not defined with

respect to a sound theoretical foundation. Moreover, they are generic and do not provide specific guidance for modeling a given domain.

How Are Classes Identified?

Several approaches have been proposed to identify potential classes, falling into three categories: linguistic analysis, patterns, and use-case and responsibility-driven approaches (Anda and Sjoberg 2005; Booch et al. 1999; Dennis et al. 2002). In addition, research on attribute and entity identification provides data-driven approaches to identifying potential classes when performing database integration (Chua et al. 2003).

Linguistic analysis involves identifying candidate classes from a written description. Many authors propose class identification through the analysis of text, focusing on noun and noun-phrase identification (e.g., Dennis et al. 2002; Graham 2001; Overmyer et al. 2001; Song et al. 2004; Stevens and Pooley 2000). According to this approach, one can "identify the nouns and noun phrases in textual descriptions of a domain, and consider them as candidate conceptual classes or attributes" (Larman 2002, p. 135). Others discuss identifying classes as abstractions that are part of the "vocabulary" of a system (Booch et al. 1999). Beyond the notion that nouns indicate candidate classes, though, the literature offers little in the way of well-justified criteria for choosing classes among these candidates. Stevens and Pooley illustrate this: "Next, we discard those which are 'obviously' not good candidate classes for any one of a variety of reasons" (2000, p. 33), offering such reasons as "outside scope," "really an event," "vague," and "time, not thing" as reasons for excluding nouns from further consideration as candidate classes.

Pattern-based approaches involve borrowing or adapting classes from similar domains (Dennis et al. 2002; Eriksson and Penker 1998). Using patterns begs the question of the quality of classes adapted or borrowed from a similar domain and does not address the question of how the classes would be identified for that other domain.

Use cases and class–responsibility–collaboration (CRC) cards provide another mechanism for class identification (Anda and Sjoberg 2005; Jacobson et al. 1999). Under these approaches, requirements are conceptualized in terms of the responsibilities required to fulfill the requirements of a use case, and are allocated to classes. According to Jacobson et al. (1999, p. 125), "every entity has to be motivated through its use in a business use case."

When Should a Subclass Be Defined?

The third stream of related literature deals with subclassification, or the decision when to define one or more specialized classes in relation to more general classes. In the data modeling context, subclassification has been proposed as useful when some instances of a class possess additional attributes, have constrained values, or participate in additional relationships relative to other instances of a class (Elmasri and Navathe 2006; Smith and Smith 1977). In the object modeling context, subclasses are claimed to be justified "when (1) the subclass has additional attributes of interest; (2) the subclass has additional associations of interest; (3) the subclass is operated on, handled, reacted to, or manipulated differently than the superclass or other subclasses, in ways that are of interest" (Larman 2002, p. 401). More generally, a new class should not be redundant with respect to (i.e., contain only the same properties as) existing classes.

Table 1 summarizes approaches to class identification and evaluation of class quality described in a range of relevant literature. In general, this literature primarily offers guidance or techniques for identifying candidate or *potential* classes for a domain, but does not offer grounded criteria for evaluating these possible classes (i.e., determining whether a potential class is a useful abstraction for a given domain) and choosing among them. An exception is the criteria proposed by Parsons and Wand (1997a), based on ideas of cognitive economy and inference. Next, we further explore the use of cognitive-based criteria as fundamental principles to evaluate the usefulness of potential classes, and guide the choice of classes such as those that might be identified through the techniques described above.

Cognitive Foundations of Classification

What Classification Is

Identifying phenomena as instances of the same class indicates they are similar in some way. In traditional cognitive approaches (e.g., Bruner et al. 1956), a class is viewed as a set of instances possessing a common set of properties (considered to be the definition of the class). An instance can be a material object, action, event, or any other phenomenon. We posit that property refers to any statement about the characteristics of an instance, including static aspects (e.g., attributes), dynamic aspects (possible changes), and rules (constraints on attributes and changes).

Table 1. Approaches to Class Identification and Evaluation

Reference	What Constitutes a Class	Identifying Potential Classes	Criteria for Subclassification
Anda and Sjoberg (2005)	Not stated	Use cases Linguistic analysis	Not stated
Booch et al. (1999)	Abstraction with clear responsibility that are supported by attributes and operations	CRC cards Use cases	Split classes with too many responsibilities Collapse classes with trivial responsibilities into larger ones
Chua et al. (2003)	Groups of attributes	Heuristics to classify or group attributes	Not stated
Dennis et al. (2002)	Set of objects sharing identical structure and behavior	Noun analysis (from use cases) Patterns	Subclasses can contain unique attributes and operations; focus on the properties that make each class unique.
Elmasri and Navathe (2006)	Entities having the same attributes	Not stated	Additional attributes Constrained values Additional relationships User-defined criteria
Eriksson and Penker (1998)	Not stated	Patterns	Only gives UML definition of generalization
Fowler and Scott (1999)	Not stated	Not stated	Not stated
Graham (2001)	Not stated	Linguistic analysis User centered analysis (focus on purpose)	Not stated
Jacobson et al. (1999)	Not stated	Use cases	Not stated
Liu et al. (2003)	Not stated	Use cases	Not stated
Overmyer et al. (2001)	Not stated	Linguistic analysis from user cases, scenarios, text	Not stated
Parsons and Wand (1997a)	Shared properties of a non-empty set of instances	Not stated	Not redundant with existing classes
Stevens and Pooley (2000)	Not stated	Noun identification	Not stated ("must be no conceptual gulf between what objects of the two classes do on receipt of the same message")

Notwithstanding differences in conceptualizing what a class means (Medin and Smith 1984; Smith 1988), there is widespread agreement that classification serves two fundamental purposes: (1) it provides cognitive economy, and (2) it supports inference. ***Cognitive economy*** refers to cognitive storage and retrieval efficiency obtained by classifying an individual phenomenon (Rosch 1978). ***Inference*** refers to the ability to infer some properties of an instance by virtue of identifying it as a member of a class, without having to directly observe these properties (Rehder and Burnett 2005; Rosch 1978; Yamauchi and Markman 2000).

The central role of classification in supporting inference provides the basis for its value. Inference entails that for a class to be useful, it is vital to be able to identify that an instance is a member of the class based on only some of the properties that define it. It follows that, by virtue of assigning the instance to the class based on some observed properties,

we can make inferences about unobserved properties (Medin et al. 2003; Ross and Murphy 1996). Classification-based inferences may affect our behavior in a variety of ways that enhance our probability of surviving. If we infer that a thing is edible, we might take action to obtain and eat the thing. Conversely, if the thing is dangerous, we might take action to avoid it.

In this context, the inferences made possible by a class ultimately determine the usefulness of the class. Clearly, if the inferences do not always hold, or might not be consequential, their usefulness is reduced. Hence, whether or not a class is useful is an empirical question. However, we claim that additional insight on class quality can be gained by considering some general principles about the way in which inferences can be supported via classification. A set of such principles is presented next.

Good Classes

Cognitive economy and inference have been used to define *necessary* conditions for a set of classes to capture the relevant knowledge about a domain effectively and efficiently (Parsons 1996; Parsons and Wand 1997a). Two conditions address effectiveness. First, to maximize the possible useful inferences about an instance once it is classified, each class should abstract the total similarity (in terms of properties) of its instances with respect to a certain purpose (instances might share additional properties not relevant to that purpose). Second, for a collection of classes to be effective, every property of interest should belong to at least one class and all properties of an instance should be derivable by considering all classes to which it belongs.

Two additional conditions address efficiency. First, every set of properties defining a class must be possessed by a nonempty set of instances. Otherwise, the abstraction is not based on phenomena that exist in the domain. Second, to avoid redundancy, each class must provide information not provided by any combination of other classes. This implies that no class should be defined to contain only the union of the properties of other classes. Such a class will require additional resources in maintenance and retrieval (either cognitively or in information systems) without providing additional inferences about instances.

The above conditions are claimed to be *necessary* for a collection of classes to be considered *acceptable* from a cognitive perspective (Parsons and Wand 1997a). However, as guidance for choosing classes, they have two limitations. First, multiple class collections can be constructed over a given domain of phenomena. The conditions offer no basis for evaluating whether any collection satisfying them is preferable to others. Second, the conditions only address inferences from class membership to instance properties, but do not take into account the benefit of identifying class membership based on partial observations about instances.

In the following, we build on the idea of inference to introduce and formalize additional principles that provide conditions for a class structure to be considered a *useful* abstraction. The key to our approach is recognizing the benefit of inference as a way to *know more about instances than is directly observed*. We then examine implications of these principles for developing conceptual models of a domain.

What is a Useful Class?

In this section, we address the question of how to limit the choice of classes in a collection to those that can be considered *useful*. We use inference-related principles to define formally what a useful class means, and use this formalization to suggest desirable relationships among classes in a class structure. Our analysis is based on three fundamental premises.

(1) A class represents instances that are similar in some way.

(2) People form classes in order to obtain some cognitive benefits. In particular, identifying an instance as belonging to a specific class based on some information enables one to infer additional information about the instance.

(3) The information about an instance at a point in time is given in terms of the *properties* it possesses at that time. This idea can be extended to properties that the instance can possess at a later time, thus introducing inferences about behavior of instances.

Since the information about instances (at any time) is given in terms of their properties, inferences will be provided as statements about properties. Moreover, given that the purpose of a class is to provide inferences about its instances, there should be similarity of inferences for instances of the same class.

We begin by defining a *domain of interest.*

> ***Definition 1*:** A ***domain of interest*** is a set of instances and a set of properties, such that each of

the instances possesses at least one of the properties and each property is possessed by at least one of the instances.

Here, the word *instances* refers to all instances that might exist in the domain at some time.

Property Precedence

To formalize class-related inferences we employ the notion of *property precedence* (Bunge 1977). Precedence means that possessing a property implies possessing another property. For example, the ability to *walk on land* implies the ability to *move on land*.

> ***Definition 2 (Precedence)***: Property p ***precedes*** property q if and only if ***every*** instance possessing q also possesses p (notation: q→p).
>
> We call q the *preceded property* and p the *preceding* (or *inferred) property*.

The properties p and q can be quite different properties related only by one implying the existence of the other. For example, paying income tax in a certain country might imply being a resident of that country. In addition, p and q can be either intrinsic to one thing, or mutual to several things. For example, a person who is at least a given age (intrinsic property) might be eligible for certain discounts (mutual properties with some organizations). Finally, precedence relationships can reflect empirical observations (e.g., all sea dwellers with fixed body temperature are mammals), natural laws, or social laws (e.g., all residents of a country are subject to income tax in that country).

The notion of precedence can be extended to combinations of properties. For example, being a senior student and having a high average grade may imply eligibility for both tuition fee exemption and for a scholarship. We formalize this as follows:

> ***Definition 3***: Let Q and R be disjoint sets of properties. A ***multiple property precedence*** (alternatively, ***multiple precedence***, ***precedence***, or ***inference***) exists between Q and R if and only if every instance possessing all properties in Q also possesses all properties in R.
>
> We use Q→R to denote precedence, and call Q the *preceded set* and R the *preceding* or *inferred* set.

It is possible that not all properties in Q are needed in order to infer every property in R. For example, assume a senior student who has high marks is eligible for tuition fee exemption and scholarship. However, all senior students might be eligible for the same. Hence, we define

> ***Definition*** 4: A multiple precedence will be called ***nonredundant*** if no proper subset of the preceded properties is sufficient to infer all preceding properties.

It is possible that a multiple precedence (whether redundant or not) includes another multiple precedence. For example, assume a foreign graduate student who has high marks is entitled to reduced tuition, scholarship and campus residence:

> Q={'foreign', 'graduate', 'high marks'}→ R={'residence', 'reduced tuition', 'scholarship'}

Assume also that every foreign graduate student is entitled to campus residence:

> Q'={'foreign', 'graduate'} → R' = {'residence'}
>
> Q'→R' can be viewed as a *component* of Q→R
>
> ***Definition 5*:** Let MP: Q→R. If proper subsets of the preceded and inferred properties exist, Q'⊂Q and R'⊂R, such that R' can be inferred from Q', Q'→R' is termed a ***component*** of Q→R.

It is possible that several components of Q→R exist that together enable inferring all information in the preceded set, R. In this case, the inference is termed *decomposable*.

> ***Definition 6*:** A set of components of Q→R will be termed a ***decomposition*** if and only if each property in R can be inferred from at least one component.

A decomposition indicates that the inference can be replaced by simpler inferences. In the example above, this will be the case if we add to

> Q'={'foreign', 'graduate'} → R'={'residence'}, the component
>
> Q"={'graduate', 'high marks'}→R"={'reduced tuition', 'scholarship'}

Not every set of components forms a decomposition. This will happen in our example, if a student has to be a foreign graduate student and have high marks to have reduced tuition.

Later, we relate decomposition to replacing a class by its superclasses. The following lemma provides a condition for non-decomposability (proofs of the lemmas and theorems, along with formal versions of the more important definitions, are provided in Appendix A.)

> ***Lemma 1***: No decomposition exists for Q→R if and only if at least one property in R cannot be inferred from a proper subset of Q.

In the student example, if this condition applied, at least one of the privileges (e.g., scholarship) must depend on being a foreign graduate student with high marks.

Proper Class

We use the inference principle to characterize the concept of a *proper class*. To do so, we first define the notion of a *potential class*.

> ***Definition 7***: A ***potential class*** is a set of properties C such that instances exist possessing all properties in C. Each instance possessing C will be termed an *instance* (or *member*) of the class.

This definition recognizes a *duality*: for a class to exist, instances must exist (at least in principle) that belong to the class. Conversely, for a set of instances to be considered a class they must possess at least one common property (Parsons and Wand 1997a). Accordingly, "class C" can indicate both the set of properties, C, and the set of instances possessing C.

Instances often possess specific values of general properties included in the class definition (e.g., a weight of 75Kg is a specific value of "weight"). To distinguish between specific values and general properties, we refer to the latter as *class-level properties*.

We now suggest one aspect of classification as a main guiding principle for determining whether a potential class is useful or not: *A class must provide information regarding its instances* ***beyond what is required to establish that an instance belongs to the class***.

This principle has two consequences. First, to have the meaning of a class, the same properties should be derivable for all its instances. Hence, these properties must be part of the class definition. Second, to make the class useful, the properties needed to determine class membership should be a *proper subset* of the full set of class properties. We call such a subset of the class properties a *base*.

> ***Definition 8***: A proper subset of class properties sufficient to identify an instance as a member of the class, where no property in the subset is inferable from others in the subset, is termed a ***base***.

Using this definition, we operationalize the value of inference with the following proposition.

> ***Premise 1*** (***usefulness***): A potential class is considered useful only if it has a base.

For Premise 1 to hold, at least one inference must exist from a proper subset of properties of the class to other class properties. Thus, we define *proper class*.

> ***Definition 9***: A potential class C will be termed a ***proper class*** if and only if at least one inference Q→R exists such that Q,R⊂C.

> ***Lemma 2***: For every proper class, a base exists.

Lemma 2 assures that a proper class is useful in terms of Premise 1. Every proper class that has inferences between subsets of properties can be specified in terms of a base (sufficient to determine class membership) and some inferences. It is possible for a given class to have more than one base. To demonstrate, consider the graduate student example. Assume class properties {'graduate', 'foreign', 'high marks', 'scholarship', 'reduced tuition', 'residence'}, and an inference {'graduate', 'foreign', 'high marks'} → {'scholarship', 'reduced tuition', 'residence'}. Furthermore, assume only graduate students with high marks can have reduced tuition fees. Then, two bases exist: {'graduate', 'foreign', 'high marks'} and {'foreign', 'reduced tuition'}. The second set is a base because one can infer from reduced tuition that the student must be a graduate student with high marks.

To distinguish between base and inferred properties, we define *derived properties*.

> ***Definition 10***: Let B be a base for class C. Properties that can be inferred from the base B will be termed ***derived properties*** with respect to B.

Since different bases might exist, different derived sets might exist as well. In the graduate student example, the derived sets are

> for {'graduate', 'foreign', 'high marks'}: {'scholarship', 'reduced tuition', 'residences'}
>
> for {'foreign', 'reduced tuition'}: {'graduate', 'high marks', 'scholarship', 'residence'}.

The existence of the second base implies that one could use, for example, university records about tuition to infer information that might not be readily available otherwise.

In summary, a potential class is specified by *all* properties that are necessary for class membership. We only require that (at least) one instance exists possessing all properties. A proper class can be specified by a base—a partial set of properties necessary and sufficient for class membership—and a set of inferences from the base to the derived properties. For such a class, there is a need to confirm that all instances possessing the base properties satisfy the inference conditions. Such confirmation can be obtained by experience or by identifying a natural or a social law that implies the inference.

Example: Assume a class is specified as {'enrolled in a university', 'graduate', 'foreign', 'high academic achievement', 'scholarship', 'eligible for campus residences'}. Consider the following base for this class: B=<'enrolled in a university', 'graduate', 'foreign', 'high academic achievement'> with the inferences

MP_1:{'graduate', 'foreign'}→{'eligible for campus residences'}

MP_2:{'graduate', 'high academic achievement'}→{'scholarship'}.

B will be a base for the class only if every foreign graduate student with high accomplishment will indeed receive scholarship and residence privileges. It is possible the second precedence will not apply universally, but will be valid for all foreign graduate students. Then if we replace the second precedence with MP_3: {'foreign', 'graduate', 'high academic achievement'}→{'scholarship'}, B with MP_1 and MP_3 will define a proper class.

Adding Properties to a Proper Class

We have not required that the class definition include *all* common properties of its instances. Thus, a class definition might not "capture" all properties common to its instances. For example, all foreign graduate students might also be entitled to out-of-country medical insurance, but eligibility for medical insurance is not part of the class definition. In such cases, the class definition can be expanded to include additional properties possessed by all instances.

Definition 11: An ***expansion*** of a class C is the union of C and another set of properties common to all class members, but not part of the class definition.

A consequence of the above definition is that expansions are also proper classes.

Lemma 3: If C is a proper class, every expansion of C is also a proper class. If B is a base of C, then B is a base of every expansion.

Moreover, we can consider the set of all properties common to all class members.

Definition 12: The ***full expansion*** of a class is the set of all properties common to all class members.

We distinguish an expansion from the full expansion because different expansions might capture different domain semantics (often described in terms of *business laws*). For example, the entitlement of foreign students for medical insurance has a different meaning than their need to have a student visa, although both are implied by being a foreign student.

While expansion applies to all class members, some class members might possess additional properties that others do not. Furthermore, these properties might be sufficient to imply class membership. For example, if only graduate students (but not all of them) are eligible for reduced tuition, then having to pay reduced tuition is sufficient to identify a student as a graduate student. We define

Definition 13: A set of properties not completely included in the full expansion of the class, but sufficient to determine class membership, is termed a ***qualifying set***.

Not all members of a class possess all properties in a qualifying set, but all instances possessing the qualifying set will be members of the class. Hence a qualifying set defines a subclass. Qualifying properties indicate class membership only for *some* instances of the class. For example, {'have feathers', 'able to fly'} is a qualifying set for birds, as not every bird can fly. A qualifying set can include some or no properties of the class. For example, assume only graduate students can work on a research thesis, but only some of them have a thesis topic. "Having a topic" is a qualifying property for being a graduate student and defines a subclass of graduate students: "graduate students with topic." This subclass is a proper class, as it contains at least one inference from the qualifying set (having a topic) to the properties not in the qualifying set (other properties of graduate students).

The following lemma indicates that, to test for a qualifying set, one needs only test whether a base can be inferred from the set.

***Lemma 4*:** A set of properties is qualifying if and only if every base can be inferred from it, and it is not a subset of the full expansion of the class.

The lemma shows that, for some instances, the qualifying set can be used to identify class membership instead of the full set of base properties. This might be more "economical" for some instances of the class, but will not work for all instances (which will have to be further examined to determine if they possess a base).

Class Structures

Class structures are collections of classes. Parsons and Wand (1997a) proposed a set of necessary conditions for a class structure to be effective and efficient. However, those conditions apply only to potential classes, not to proper classes. Here, we extend the analysis to proper classes. In particular, we extend the concept of effectiveness to include inferences, and efficiency to mean eliminating redundant inferences. In the following, we assume every class structure fulfils the following conditions:

(1) every property of interest is included in at least one class definition, and

(2) each property of every instance belongs to the definition of at least one class in which the instance is a member.

Class structures are often extended by deriving subclasses from existing classes. We note

***Lemma 5*:** Every subclass of a proper class is a proper class.

We now propose a particular kind of nonredundant class structure.

Definition 14: A collection of proper classes will be termed ***efficient*** if every class in the collection provides at least one inference not provided by any of its superclasses.

This definition entails that, in an efficient collection, inferences will always be expressed at the highest possible abstraction level. It leads to two guidelines when examining a collection for possible addition or removal of classes. Both suggest that a class should not be included unless it provides inferences beyond those already existing in a set of classes. One guideline covers the case when a class should be removed, and the other the case for adding a class.

Guideline 1 (Minimality): If all information provided by a proper class can be inferred from proper superclasses, then the class should be removed.

Guideline 2 (Nonredundancy): Given a collection of proper classes, a new class may be added only if it provides at least one inference not provided by any classes already in the collection.

These two guidelines combined indicate how a class structure can be changed to become efficient and stay efficient under further changes. First, it should be made minimal. Then, if a new class is considered for addition to the structure, this should be done as follows:

(1) If the candidate class is a subclass of existing classes, it should provide a new inference.

(2) If the candidate class is a superclass of an existing class, it should be added only if, together with other superclasses, it can provide the same information as an existing subclass, which should then be removed from the collection.

(3) If the class is neither a subclass nor a superclass of existing classes, it should be added only if it provides new inferences, or enables by its addition the removal of other classes. (This case might require additional considerations, such as ease of inference, which we do not consider here.)

The following analysis assumes that we deal only with efficient class structures of proper classes, and that any change made should result in a collection that remains efficient.

In an efficient class structure, a class should provide inferences not derivable from its superclasses. This means that no class can be decomposed without loss of some class-related inference. The following lemma formalizes this requirement:

Lemma 6: In an efficient class structure, no class possesses a base and a matching inference that is decomposable.

In the following, we present three theorems that provide necessary conditions for adding a new class to an efficient class structure. The theorems are expressed informally, to emphasize their intuitive meaning and their practical use (see Appendix A for formal versions).

***Theorem 1*:** For a subclass to have a new inference and be non-decomposable, it must either have at

least two new properties or include a qualifying set for the class.

This theorem provides two conditions, at least one of which *must* hold, for a subclass to be added. For example, if "graduate student" is formed by indicating a graduate program of registration for a "student," there should be additional properties (e.g., thesis topic or advisor). Alternatively, having a thesis topic qualifies a student as a graduate student.

Several mechanisms are commonly used to add new classes, such as intersecting the membership of two existing classes (defining a class by a union of their properties), and limiting the values of some properties. An example of the first mechanism is forming a class of "taxpaying seniors" from "tax payers" and "senior citizens." An example of the second mechanism is forming a class of "senior citizens" based on "age 65 or higher." We now analyze when each of these class forming approaches might yield new useful subclasses.

First, consider the union of two class definitions. The following theorem establishes conditions for when it might be useful to add a proper class defined by the union of properties of two proper classes (each of which will be a superclass of the new class):

> ***Theorem 2***: For a class defined by the union of properties of two classes to have a new inference and be non-decomposable, it must have at least one new property or include at least one subset of properties which is qualifying for at least one of the original classes.

The theorem implies two conditions, at least one of which must hold, to add a class defined by the union of properties of two classes. For each condition there is a seeming (cognitive or information processing) benefit. In one case, the union class provides additional information. In the other, less information is needed to identify an instance as a member of the new class than as a member of each class separately. An example for the first case is a class formed from foreign residents and students ("foreign student"). In this case, it will be useful if members of this class have some properties in addition to those of either class (e.g., a requirement to have special health insurance). An example for the second case is sea mammals. Mammals have fixed warm body temperature and mammary glands. Sea dwellers can swim, spend all their life in the water, and can dive. However, it is sufficient to know an animal is warm blooded (fixed temperature) and lives only in the sea to identify it as a sea mammal.

Consider now forming classes by limiting the values of a class-level property (e.g., all citizens defined by "age" of 65 or more). We distinguish between a class-level property which is a property in the definition of the class (e.g., age or address), and the ***manifestation*** of the property for an instance. For example, a manifestation of being able to move on land is being able to walk or to hop. A special case of manifestation is a value (e.g., a specific age or address). Let p be a class-level property in C and denote by C/p=v the set $C \cup \{p=v\}$; namely, a set of properties together with a specific "value" for one of them. C/p=v defines a subclass for which p=v is one of the class-level properties (it is common to all instances of the new class). For example, let one of the class properties of students be "program of study," then a possible subclass is Student/'program of study'='counseling'.

The following theorem suggests when it might be useful to add a proper class defined by limiting the value of a property.

> ***Theorem 3***: For a subclass defined by a value of a property to have a new inference and be non-decomposable it must either have at least one additional property, or include a qualifying set for the class where p=v is included in this set.

The theorem implies two conditions, one of which must hold, to add a class defined by limiting the value of a property. Either the new subclass provides additional information or less information is needed to identify some instances of the original class. For example, each student in the subclass formed by limiting program to "counseling" might have an assigned practice site. Alternatively, knowing that the student is in a counseling program might be sufficient to know s/he is a graduate student.

To summarize, one of two underlying reasons to add a subclass should exist. First, the subclass provides additional information to that of its superclasses. This was the case for the counseling academic program example. Second, the subclass provides a more efficient way to identify some of the original class members than by simply applying the mechanism used to form the subclass (e.g., testing that an instance belongs to both ancestor classes). This was the case of "sea mammals." These conditions are outcomes of the requirement that added classes should provide new inferences which are not provided by their superclasses.

Finally, although we have discussed the reasons for adding subclasses, we have not discussed the justification for adding superclasses. This might be considered when the analysis of the domain is performed "bottom up," recognizing higher levels of abstraction. Such levels can be recognized when a class is decomposable—that is, it can be substituted by "simpler" versions. For example, assume that being age 60 or older, resident of a city, and of low income entitles a person

to discounts in property taxes, no fee for city recreational services, and bus discounts. An analysis might show that all city residents whose age is 60 and above receive bus fare discounts and free recreational services, while all those who are city residents with low income are entitled to tax discounts. In such a case, the superclasses "senior residents" and "low income residents" should be considered. Our guideline of minimality provides direction in this case. Adding superclasses should be considered if the inferences they provide can replace those provided by subclasses (without loss of property and instance information).

Application to Systems Analysis

The previous section suggests an approach to understanding the usefulness of individual classes and collections of classes based on principles that explain why and how humans make sense of the world by classifying phenomena. The key to the approach is in recognizing the importance of inferences from some properties to others. Inferences capture the type of conclusions and actions that can be taken based on certain properties of members of classes. However, in a given situation, not all possible information may be of interest. The choice of inferences provides a context for the phenomena of interest. For example, suppose that foreign graduate students are eligible for special medical insurance and for a visa allowing a spouse to work in the country. Only the former might be of interest to a university. Hence, identifying the appropriate inferences is a way to capture the reason for choosing classes in a given context.

We now explore how the proposed framework can be used in information systems analysis. We focus on the development and evaluation of conceptual models of a domain. Our analysis is guided by the view that considering the inferences provided by classes is critical to all classification-related activity. Such inferences often result in derived, nonobvious properties. We argue that classes designed by following our approach will be semantically clearer and more easily understood by humans and, therefore, more appropriate for conceptual modeling than classes developed without following the approach. In information systems design, such implied properties may be represented in data, calculated by software, or derived by manual procedures.

Conceptual Modeling

One of the key tasks in developing a conceptual model of a domain is identifying a set of classes that describe the things of interest. As discussed earlier, existing approaches provide guidance for identifying potential classes using techniques such as linguistic analysis, use case analysis, and responsibility-driven approaches. However, these techniques provide very little well-grounded guidance on evaluating the quality of potential classes generated using them. The framework proposed above provides a set of criteria for determining whether a proposed class can be useful in the context of an application. This framework complements existing approaches by providing clear rules for evaluating potential class in terms of their inference value, and for identifying refinements to class structures (e.g., adding classes or deleting existing classes) to capture additional relevant inferences and avoid redundancy.

Recall that a class implies *commonality of properties and inferences*. While properties are usually explicit (e.g., as data), inferences are not usually manifested explicitly. For example, the properties (Name, Address, Weight, DateofLastInoculation) are not enough to distinguish between the classes of humans, pets, and livestock. Instead, inferences might be embedded in software or applied when the data (resulting from a query) is used by humans.

We now suggest specific rules for choosing classes in conceptual modeling. We present the rules as questions to be asked by the modeler when developing or examining classes in a class structure. They are intended to refine a given set of proposed candidate classes (where properties are represented as attributes and relationships) and are of three types. First, a ***screening*** rule establishes a necessary condition for a given potential class to be useful. Second, ***nonredundancy*** rules are used to evaluate the usefulness of proposed subclasses of multiple superclasses with respect to the superclasses. Third, ***formation*** rules are used to evaluate whether potential subclasses identified using commonly proposed mechanisms might be useful.

Most of the proposed rules address sub- and super-classification relationships among classes, since this is a major consideration in forming class structures. Because a good class structure should assure coverage of all relevant aspects of the domain, and reduce redundancy, whether a class should be included or not depends on the other classes in the class structure. The relevant relationships among classes for this purpose reflect sub- and super-classification. Hence, guidance on the composition of class structures is closely tied to these relationships.

To illustrate the application of the rules and the implications of following them, we analyze a published example conceptual model that was developed without explicitly following

our approach. Figure 1 is a conceptual schema adapted from a popular database textbook (Elmasri and Navathe 2006). In the original, the example was used to illustrate various extended entity–relationship modeling concepts. Our adaptation changes the notation of the original to a visually simpler one in which the attributes of classes are placed inside the box denoting the class (rather than linked to attribute names in ovals). Also, parts of the original diagram not relevant to our rules are omitted. Finally, we make reasonable extensions by adding a subclass EMPLOYEE of PERSON, a subclass TEACH_ASST (hereafter T_A) of EMPLOYEE and STUDENT, and an indication that INSTRUCTOR_ RESEARCHER (hereafter I_R) is a subclass of EMPLOYEE.

Finally, recall that in our model a class is defined by a set of properties that all of its instances must possess. According to Figure 1, all of the instances of I_R possess <supports> and <teaches> and must also possess the properties of either FACULTY or GRAD_STUDENT. However, this does not provide one unique set of defining properties. Hence, I_R is not even a potential class according to our definition. Therefore, it cannot be a proper class and, as presented in Figure 1, cannot be part of a class structure. To remedy this situation and still maintain the constraint that an I_R must be either FACULTY or GRAD_STUDENT, we can define additional subclasses, FACULTY_INSTRUCTOR (F_I, a subclass of FACULTY and I_R) and GRAD_INSTRUCTOR (G_I, a subclass of GRAD_STUDENT and I_R), as explained below.

In the following, we present a set of classification rules and apply them to the example in Figure 1. The schema resulting from applying these rules, as well as enforcing the requirement that a schema contain only proper classes, is shown in Figure 2.

Screening Rule

Rule 1: A proper subset of the properties of a potential class that is sufficient to determine that an instance belongs to that class must exist.

This rule applies the definition of proper class and Lemma 2 stating that for every proper class, a subset of the properties (a base) is sufficient to identify class membership. The rule serves as a filter to ensure that only proper classes are included in a conceptual model. If Rule 1 is not satisfied, the potential class is not useful in the sense of enabling inference. Thus, Rule 1 supersedes the remaining rules. If Rule 1 is satisfied, a base exists and an inference can be made from that subset to the remaining properties. An instance can then be identified as a member of the class based only on a subset of its properties.

Example: We analyze the classes in Figure 1 for conformity to Rule 1. Since we do have the original information that led to Figure 1, we consider conditions that must be true for the rule to hold. For example, PERSON is defined by the properties {SSN, Name, Address, BDate}. Assuming that possessing SSN implies possessing the remaining properties of PERSON, Rule 1 is satisfied. Similar reasoning can be applied to the other classes. In addition, any subclass containing at least one property in addition to its superclass(es) satisfies Rule 1. This is the case for all proper subclasses in Figure 1. (However, this does not mean that every subclass conforms to the remaining rules.) Thus, Rule 1 provides guidance in determining the potential usefulness of classes derived by methods such as noun-phrase analysis or use case analysis.

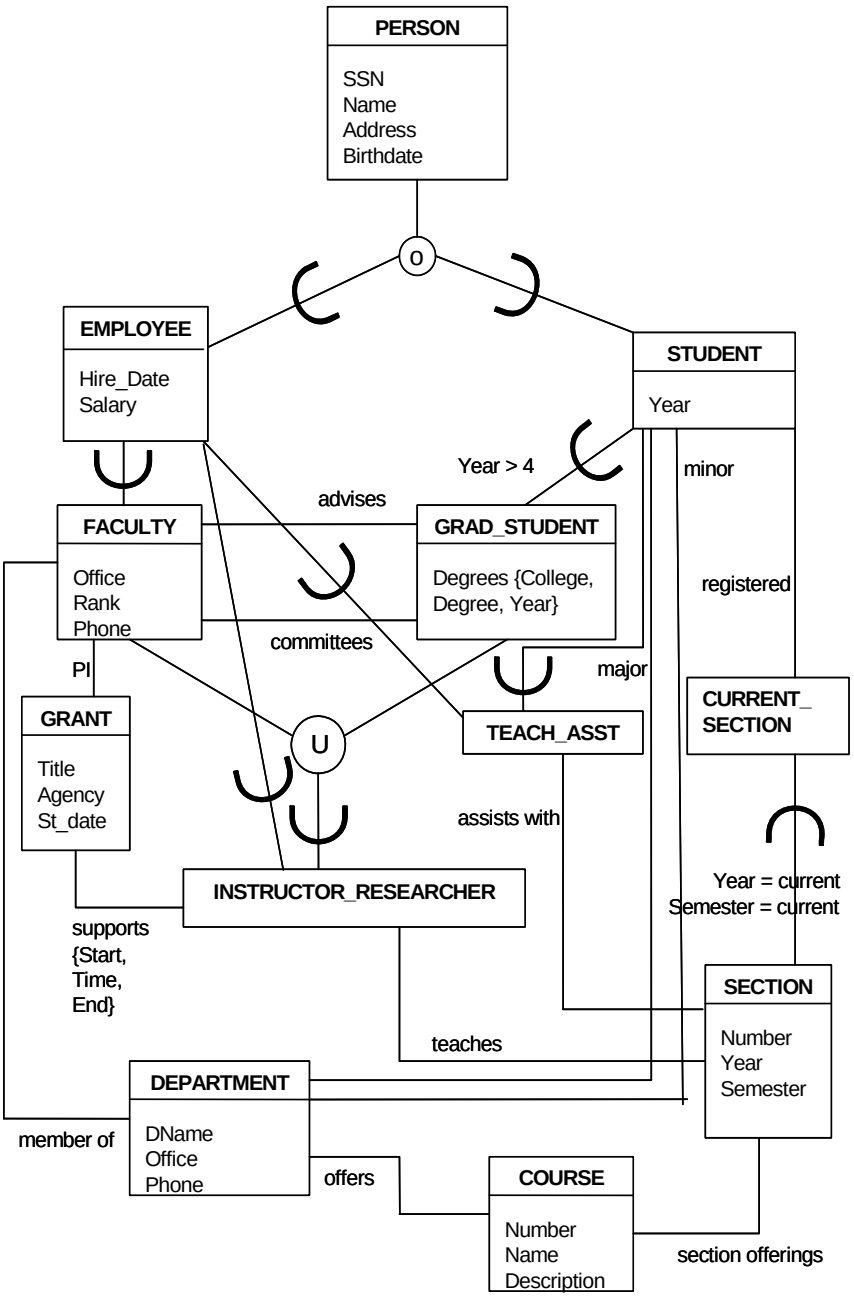
Nonredundancy Rules

Based on Theorem 1, for a candidate subclass to have a new inference, at least one of the following two rules must hold. These rules assure that a proposed subclass provides either an alternative way to identify some members of a superclass, or an additional inference.

Rule 2: A candidate subclass must possess qualifying properties for the superclass.

This rule applies one condition of Theorem 1. Qualifying properties indicate that instances possessing them belong to the superclass, even though the properties are not part of the superclass definition (thus, not all instances of the superclass possess these properties). A qualifying set indicates that a useful subclass of a class (containing the instances of the latter that possess the qualifying properties) might exist, and may be included in the conceptual model if the additional inference it provides is deemed useful to the application. The subclass might be useful because its members can be inferred to be members of the superclass based on "optional" (with respect to the superclass) properties.

Example: In Figure 1, consider the subclass GRAD_ STUDENT of STUDENT. It has the property Advises linking it to FACULTY. If only graduate students can be advised by faculty, than graduate student is potentially useful and Advises is qualifying (but not necessary) for a student. In contrast, from a commonsense semantics point of view, the property Degree is likely not a qualifying property, as we would expect FACULTY also to possess it (although this is not shown in Figure 1). Likewise, Advises is qualifying to FACULTY with respect to EMPLOYEE.



Notation: An "o" in a circle connecting subclasses with a superclass means that the subclasses are overlapping (i.e., instances of the superclass may belong to both subclasses). The symbol "∪" on a line indicates a superclass/subclass connection. The symbol "∪" in a circle indicates a union of the instances of two classes.

Figure 1. Partial Conceptual Schema for a University Application (Based on Elmasri and Navathe 2006, p. 120)

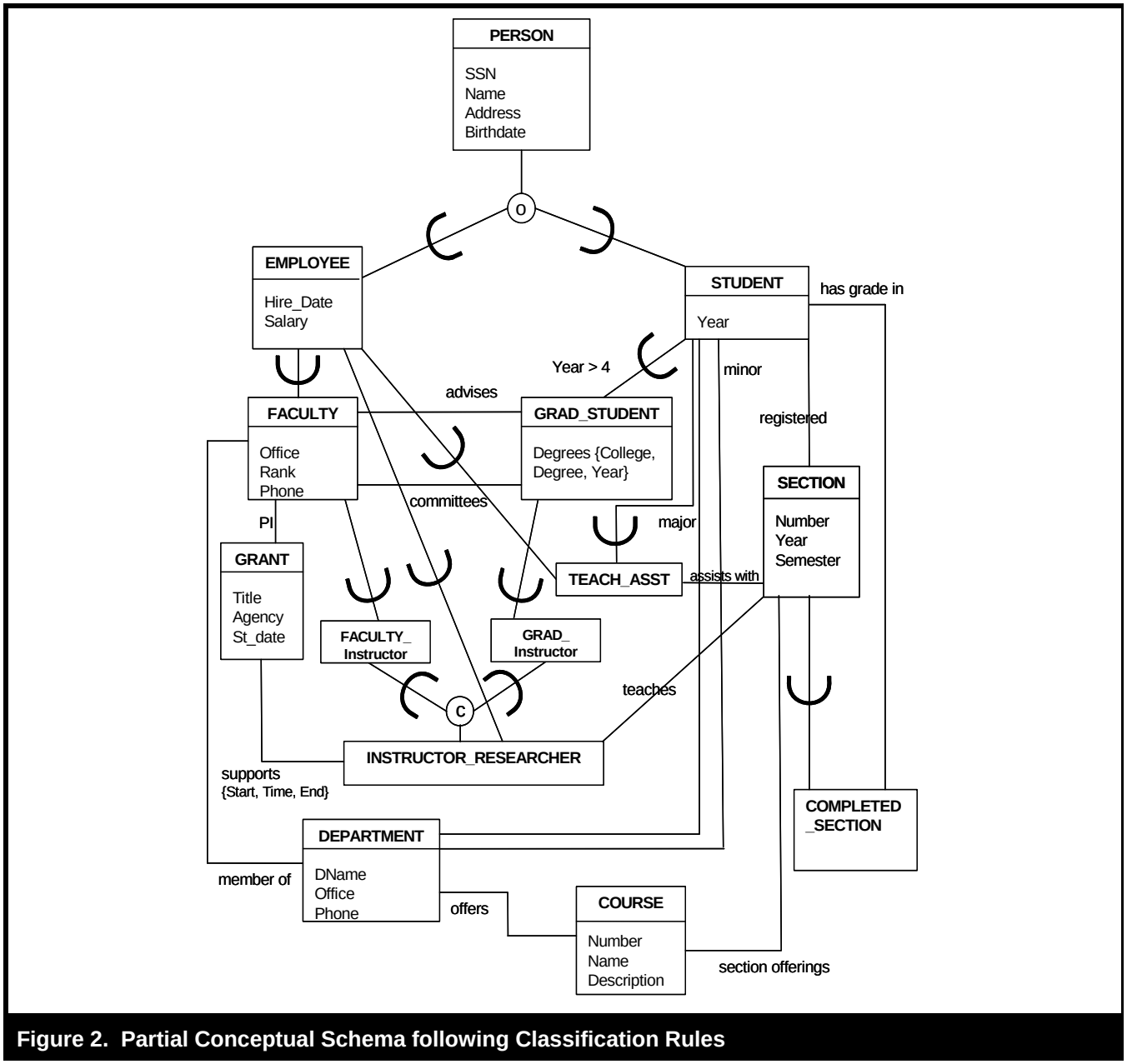


Figure 2. Partial Conceptual Schema following Classification Rules

Figure 1 also shows that an I_R must be either a FACULTY or a GRAD_STUDENT but, as indicated earlier, this is not a proper class. Figure 2 presents a modified schema that defines additional subclasses: F_I (a subclass of FACULTY and I_R) and G_I (a subclass of GRAD_STUDENT and I_R). Whether such potential classes are useful is determined by whether they adhere to the rules proposed here. In the absence of additional information about this domain, we cannot answer this question definitively. However, Figure 2 suggests that we need to ask additional questions about the domain to clarify whether or not these subclasses would add information. For example, it might be that G_Is might qualify for a tuition exemption and F_Is receive credit toward teaching load. Thus, our approach provides guidance for additional questions that need to be asked. Figure 1 does not provide any basis for understanding why we care to model that an I_R must be either FACULTY or GRAD_STUDENT. In contrast, Figure 2 requires us to inquire whether and why these classes provide useful inferences.

Rule 3: A candidate subclass of a class must have at least *two* additional properties to those of the superclass.

This rule applies the other condition of Theorem 1. It might appear counterintuitive to the common view that a subclass can be defined if instances have at least one additional property. However, the rule is a result of the condition that, to be useful, a potential subclass must have at least one additional inference. If the subclass does not have qualifying properties, it must have at least two additional properties.

Example: Subclasses in Figure 1 include EMPLOYEE and STUDENT (of PERSON). Both subclasses have at least two properties in addition to the superclass, and thus satisfy the rule. Another subclass in the model is CURRENT_SECTION (of SECTION). This class possesses an additional (intrinsic) property, restricted values of Year and Semester. In addition, it possesses an additional (mutual) property, <registered> (with STUDENT), and appears to satisfy Rule 3. However, if registration histories or registration in future sections are allowed, being registered would be a property of all sections and, therefore, not a qualifying property for SECTION. Thus, without additional properties, this class does not satisfy Rules 2 or 3.

Formation Rules

Two common approaches for creating classes are by taking the union of the properties of two or more existing classes (the instances are the intersection of the instances of the original classes), and by restricting the value of properties of instances. Based on Theorems 2 and 3, we identify two rules for each approach, at least one of which must hold for a class created with the approach to be considered useful. Rules 4 and 5 deal with subclasses of multiple superclasses.

Rule 4: A subclass of multiple superclasses must permit the identification of its instances using fewer properties than are required to identify it as an instance of all superclasses.

If Rule 4 is satisfied, the subclass provides a simpler way to identify instances belonging to multiple superclasses.

Example: In Figure 2, we introduced the subclasses F_I and G_I to capture the constraint that I_Rs must be either FACULTY or GRAD_STUDENT and speculated that these subclasses might reasonably have additional properties. For example, assume only faculty members who teach receive a special allowance for buying textbooks. Having such an allowance will identify an employee as both a faculty and an instructor. Thus, an instance can be identified as an instructor using fewer properties than needed to identify it as both a faculty and an instructor.

Rule 5: A potential subclass of multiple superclasses must possess properties in addition to all superclasses.

If knowing that an instance belongs to a subclass implies knowing more about it than simply that it belongs to all its superclasses, such a subclass can be useful. Note that having at least one additional property with respect to all superclasses means that instances of the subclass have more than one additional property with respect to each superclass. Thus, Rule 3 is satisfied.

Example: Consider the class T_A as a subclass of both STUDENT and EMPLOYEE. T_As <assist with> SECTIONs. This (mutual) property is not one of the properties of the two superclasses as not all students and not all employees assist with classes.

If neither Rule 4 nor Rule 5 is satisfied, the candidate subclass offers neither inference value nor cognitive economy (instead, there is a cognitive "diseconomy" associated with storing an additional class). Therefore, it is not useful according to the inference requirement.

Similar to Rules 4 and 5, Rules 6 and 7 are based on Theorem 3 and cover subclasses created by property restriction.

Rule 6: A potential subclass based on restricting the value of properties of a superclass must provide a qualifying set for the original class.

In Figure 1, CURRENT_SECTION is defined by Qtr=Current & Year=Current, which restricts the values of properties of the class SECTION. However, such a restriction might not be unique to the current section. For example, in the university context, it could refer also to teaching assignments of faculty. Hence, it cannot be a part of a qualifying set for sections.

A subclass formed by value restriction provides a new inference if the following holds.

Rule 7: A potential subclass based on restricting the value of properties of a superclass also must possess at least one additional property.

Rule 7 does not contradict Rule 3, as a property value amounts to an additional property.

Example: CURRENT_SECTION possesses the mutual property <registered> with STUDENT. However, this property would also likely apply to SECTION. Thus, CURRENT_SECTION also fails the usefulness test of this rule. In fact, a more compelling case could be made to model COMPLETED_SECTION as a subclass of SECTION, since instances would have additional mutual properties with STUDENT, such as Grade. Figure 2 shows this revision.

We briefly summarize what the rules imply with respect to a set of classes. First, according to Rule 1 every class must be a proper class (hence a potential class). Second, any class specializing an existing class must satisfy at least one of Rules 2 and 3. For subclasses of multiple superclasses at least one of Rules 4 and 5 must hold. Finally, if a subclass is defined by limiting the values of a property of another class, it must satisfy at least one of Rules 6 and 7. By applying these rules to Figure 1, we find that some classes do not adhere to all the rules and thus do not seem to be useful (e.g., CURRENT_SECTION), while others are misspecified (e.g., I_R is not a proper class). Figure 2 proposes an alternate schema containing only proper classes that adhere to the modeling rules. The example demonstrates how the rules can be used to evaluate the quality of proposed classes and to guide the development of conceptual models.

Finally, note that, even when a rule specifies that additional properties need to exist, they do not have to be specified explicitly as attributes of the class. Instead, they may be embedded in software or inferred by users. That is, the rules are intended to prevent the inclusion of classes that do not provide inferences in addition to those provided by other classes. However, not all inferred properties identified by the rules will necessarily be included within a database design. A designer might choose to store inferred properties in the database, or might (1) include their calculation within the software (e.g., calculating a tuition fee discount for graduate students who have an approved thesis topic), or (2) exclude them from the software design altogether, leaving their determination to humans based on stored data (e.g., issuing an invitation to an event based on age and residence of a person). Identifying inferred properties makes such design decisions more explicit and thereby can serve to clarify (i.e., provide justification for) the requirements and specifications of information systems.

Evaluation of the Proposed Rules

We have proposed rules to guide the identification of classes in conceptual modeling based on the role of inferences in classification by humans. An important issue for the practical usefulness of the rules is whether the differences in models produced by following (or not following) them facilitate (or inhibit) understanding and communication about the modeled domain. To explore this question, we conducted an empirical study by inviting a panel of experts to evaluate the differences between the models shown in Figures 1 (original) and 2 (modified so that all classes are proper classes and conform to the rules presented in the previous section).

Participants and Task

The panel consisted of 10 modeling and domain experts: five academic colleagues with considerable research and teaching experience related to conceptual modeling; four Ph.D. students in information systems with research and/or practical experience in conceptual modeling, and one practitioner with extensive experience in information systems modeling. Modeling experts were chosen because they were expected to be familiar with the reasoning that takes place when constructing a model (and thereby able to contribute to a meaningful analysis of the models), sufficiently motivated to provide thoughtful responses, and familiar enough with modeling notations to require only general instructions in order to complete the task.

Participants were asked to follow the instructions in Appendix B with respect to the diagrams in Figures 1 and 2. The objectives were (1) to focus participants' attention on the differences between the diagrams, (2) to ask participants to formulate questions that could be asked to understand why the differences arose, and (3) to get a sense of whether participants considered one representation to be more useful or realistic than the other or not and, if so, why. Participants were not informed about the rules used to generate the diagrams.

Results

As discussed in the previous section, there were two important differences between the diagrams. First, Diagram 1 proposed a subclass of sections consisting of those offered in the current semester, and linked this class to a student class via registrations. However, without additional properties, this subclass violated the rule requiring that subclasses based on restricting the values of properties of the superclass must possess at least one additional property. In Diagram 2, this subclass was eliminated and students were linked directly to sections via a registration relationship. A new subclass of the section class was created to capture completed sections for

which students have been assigned a grade. Second, Diagram 1 used a "union class" (in the example, a class consisting of instances that were a subset of the instances of two other classes) to capture the constraint that instructor-researchers must be either faculty or graduate students. In contrast, Diagram 2 captured this constraint while adhering to our modeling rules by including two subclasses of instructor-researcher, one to indicate those who were also faculty and the other to indicate those who were also graduate students.

All participants identified these two issues as the key differences between the diagrams. With respect to the treatment of SECTION and its subclasses, the results reveal a clear preference for the representation in Diagram 2 over that in Diagram 1. *Nine of the ten respondents* expressed a preference for Diagram 2 on the grounds that it reflected more realistic semantics based on their knowledge of the domain, while the remaining respondent did not express any preference between the diagrams. Appendix C summarizes the respondents' feedback on this difference. For example, R2 stated that Diagram 2 "helps to make some additional inference, such as a grade can be given only to students who have completed a course." This explicitly recognizes the role of inference in classification and seems to reflect an implicit application of the notion of a qualifying property as expressed in Rule 2. Similarly, R4 argued that "in the typical university, students should have registrations associated with all sections they have taken" in supporting the conclusion that Diagram 2 is more appropriate in this case. We have added in the tables of Appendices C and D comments indicating the rules that we identified as implied by, or related to, the comments made by participants.

With respect to the treatment of I_R, there was also a clear preference for modeling explicit subclasses F_I and G_I to capture the constraint that an instance of I_R must belong to one of these other classes. Six of the seven respondents who expressed a preference chose the representation in Diagram 2 that reflected changes made to Diagram 1 in order to conform to the proposed modeling rules. Appendix D summarizes the respondents' feedback on this difference. For example R5 noted that "Diagram 2 allows for differences between grad instructors and faculty instructors and suggests that a person could be both a faculty member and a graduate student. This describes my last year in my Ph.D. program." Likewise, R8 finds that "Diagram 2 seems more useful; it clarifies the structure of the discussed domain instance in places where it might differ from similar instances of the domain."

One respondent (R7, a Ph.D. student who had considerable experience as a software developer) indicated a preference for Diagram 1 on the grounds of simplicity (R1, an academic, also noted that Diagram 1 was simpler in the treatment of I_R). However, R7 also indicated he could see why differences would exist and, if these were important, then it was useful to have the classes. Three respondents indicated no preference between the diagrams regarding this difference. However, two of these respondents indicated that, based on Diagram 2, they would ask why the subclasses F_ I and G_I are needed (the third identified the difference but focused questions and comments exclusively on the treatment of SECTION). These responses indicate that the representation based on the modeling rules invokes appropriate questions to clarify semantics of the domain that might not be explicitly expressed in the diagram.

To examine the responses in greater detail, we developed a set of codes for organizing the respondent comments that favored Diagram 2 over Diagram 1. These codes indicate high level descriptions of statements providing support for one diagram over the other. Based on respondents' comments, three codes were developed for each of the two primary differences between the diagrams. With respect to the treatment of SECTION, the first code focused on whether or not student registrations in past and future sections (versus only current sections) could be modeled. An example statement assigned to this code is "It is possible in D2 to see in which completed sections students were registered; this information is lost in D1" (R7). The second code dealt with the related issues of keeping track of grades that students received in past courses, covering such statements as "In D2, students have grades only in completed sections; D1 says nothing about grades" (R10). The third code covered questions that might be asked to determine whether the classes COMPLETED_SECTION (Diagram 2) or CURRENT_SECTION (Diagram 1) were useful, with a sample statement being "Is it the intent of the system to keep historical records for which students have taken which sections?" (R1).

Similar codes were developed for the treatment of I_R. First, the use of a "union class" in Diagram 1 means that I_R is not a potential class. Instances of it do not have a fixed set of attributes, as they could have either the attributes of FACULTY or of GRAD_STUDENT. Second, by using subclassification, Diagram 2 requires that an I_R must be either a FACULTY or GRAD_STUDENT, while this is not explicit in Diagram 1. Third, a code was used to categorize questions that might be asked to determine if the subclasses F_I and G_I were needed.

One of the researchers reviewed the respondent comments and coded appropriate statements using the above coding. The raw comments and codes were also provided to a second coder (a Ph.D. student in information systems who was not

Table 2. Coded Respondent Comments Supporting Revised Diagram

Code	Coder One Respondents (R#)	Coder One Count	Coder Two Respondents (R#)	Coder Two Count	Agreed Statements
S.1: Keep track of student registrations for future (or past) sections (as Diagram 2 permits) versus only current sections (as Diagram 1 permits).	R1, R4, R6, R7, R9, R10-1, R10-2*	7	R1, R4, R6, R7, R9, R10-1, R10-2*	7	7 of 7 (100%)
S.2: Keep track of grades that students received in past courses (Diagram 2).	R1, R2**, R4, R5, R6, R7, R9, R10	8	R1, R2**, R4, R5, R6, R7, R9, R10	8	8 of 8 (100%)
S.3: Questions that might be asked about Student (or Section) to determine whether Completed_ Section (or Current_Section) are needed.	R1, R2, R3, R5, R6, R7, R8, R9, R10	9	R1, R2, R3, R5, R6, R7, R8, R9, R10	9	9 of 9 (100%)
IR.1: In Diagram 1, the attributes of I_R are not fixed (since it might be either FACULTY or GRAD_ STUDENT), while in Diagram 2, I_R has a clear, fixed set of attributes.	R1	1	R1	1	1 of 1 (100%)
IR.2: Distinguishing FI and GI as subclasses of both FACULTY and GRAD_STUDENT provides clearer domain semantics indicating that an I_R must be either a FACULTY or GRAD_ STUDENT.	R2, R5, R6, R8, R10	5	R2, R5, R6, R8, R10	5	5 of 5 (100%)
IR.3: Questions that might be asked to determine if FI and GI are needed.	R1, R2, R4, R5, R6, R7-1, R7-2, R9	8	R1, R2, R4, R5, R6, R7-1, R9	7	7 of 8 (88%)

*Rn-1 and Rn-2 indicates that there were two separate statements from Respondent 10 regarding this category.

**Indicates that Coders 1 and 2 tagged different statements that were later agreed on as being semantically equivalent.

otherwise involved with the research), who independently coded the respondent comments.

As Table 2 shows, there was near unanimous agreement between the coders on the assignment of statements to categories. Furthermore, there was strong agreement among respondents that, relative to Diagram 1, Diagram 2 offered a preferred approach to modeling sections by distinguishing past sections from all sections, thereby allowing the representation of student grades. There was somewhat less agreement on the preference for Diagram 2 with respect to the treatment of I_R. However, of the seven respondents who addressed this issue, six indicated that the semantics were clearer in the revised diagram. More importantly, in response to "What questions might you ask to clarify the reasons for these differences," seven of the ten respondents explicitly indicated that they would ask why the classes were needed, or what distinguished them from their superclasses.

Collectively, these findings offer compelling preliminary evidence that the proposed modeling rules generate models that have clearer semantics and are preferred by modeling experts. However, we recognize that this evaluation is necessarily limited. Our research focused primarily on developing an artifact (model and method to support the construction of useful class structures) in an area lacking well-grounded design guidelines. Given the extent of that analysis and space limitations, the empirical evaluation is intended and able only

to demonstrate that the artifact produces conceptual models perceived as comparatively more useful than those produced using extant approaches.

Conclusions and Future Research

This paper formalizes ideas underlying effective classification, based on the cognitive principle that a class should be useful in providing inferences and should provide additional inferential value with respect to other classes in a collection. We derive several properties of classes constructed according to this principle. Based on our analysis, we propose seven practical rules to guide the development of conceptual models that specify classes of phenomena. The rules can assist a systems analyst in identifying whether a class is useful when modeling a domain, where usefulness is defined in terms of whether inferences can be made based on class membership. We demonstrate how the rules can be applied using an example. In addition, we provide empirical evidence from a panel of experts that the rules can guide the construction of semantically clearer and more useful models. Since the rules are based on the notion that a class is an abstraction of instances in a domain, we believe the method is applicable to other information systems contexts involving the identification of classes, including semantic data modeling, object-oriented analysis, and ontology development.

Our work follows the design science guidelines proposed by Hevner et al. (2004). We proposed a design artifact consisting of a model of good classification structures and a method (rules) for constructing such structures. The artifact addresses an important issue in the management of information systems development for which theory-based guidance is currently lacking: developing high quality conceptual models of application domains. In this context, identifying "good" domain classes can help better reflect users' views and lead to the design of information systems that better support these views. We developed the artifact using a formal modeling approach and evaluate the resulting rules empirically. The formal approach consisted of modeling to derive logical consequences (theorems and lemmas) of a core foundation and to translate these results into a practical method (modeling rules). The empirical evaluation involved two steps. First, we chose a recent published example as an indication of current practice and to reduce potential researchers' bias, and created an alternative conceptual model by applying the proposed modeling rules to the example. Second, we collected evidence from a panel of experts showing that the revised conceptual model was preferred to the original, thereby providing a bridge between design science and natural science (March and Smith 1995). The research contributions include theoretically derived rules to guide conceptual model development, established rigorously via formal proofs of useful results related to the evaluation and choice of concepts used in modeling phenomena in a domain, as well as novel insights with respect to class quality in information systems analysis. The rules are anchored in a theory derived from psychological principles humans use to survive and adapt by classifying phenomena. The research includes both technical presentation and practical framing in terms of application to an important problem in systems analysis and design.

Our results open the door for several avenues of research. First, the insights provided by the panel of experts suggest there is a need to explore more fully in controlled experiments and in field settings the empirical value of developing conceptual models in which the classes defined adhere to the framework proposed in this paper. Some specific questions of interest include: Are such conceptual models easier to understand and use? Can the principles be used in the practice of information systems development, and how easy will they be to use? Will the use of the principles lead to designs (based on the knowledge included in the conceptual model of the application domain) that are demonstrably more effective than traditional approaches? Can the rules be used directly to guide the choice of classes (e.g., as types in entity–relationship models or classes in UML class diagrams) in the design process? Such questions are amenable to traditional behavioral science research in the information systems field. Research on these questions can include model creation (writing) experiments (with experts and novice subjects) and model interpretation (reading). Both types of experiment can use outcome measures (quality of models or level of understanding) or process measures (e.g., obtained via process tracing). In addition, since we did not find violations of all of the modeling rules in the example used with the expert panel, the remaining rules also need to be evaluated empirically.

Second, we believe the class evaluation and choice principles proposed in this research are applicable to other information systems domains yet to be explored. For example, recently there has been great interest in developing "domain ontologies" to support information sharing and interoperation in semantic web applications. A domain ontology is essentially a formal specification of a set of classes and their relationships. However, there is very little research dealing with methods to design useful ontologies. The principles proposed here can provide such guidance. Similarly, we believe the principles are applicable in a number of other areas, such as the organization of menu structures in application software, as well as in the design of repositories that effectively categorize documents, including e-mail.

Third, we have focused on classifying "things" (typically considered entities in database design). However, other types

of phenomena are subject to classification, such as information in user interfaces, business rules, actions, or business processes. It would be of interest to explore how the principle of inference can inform classification and organization of such phenomena.

Fourth, the approach can be embedded in an automated system to guide the development of conceptual models by evaluating identified classes and recommending changes based on the rules. Such a system could also force analysts and users to probe for refined definitions of classes that take inferences into account. The development of such a system may also suggest enhancements to the proposed rules, and more detailed guidance for their use in practice.

Finally, our analysis has not considered the "cost" of observing base properties and inferring the other properties. Whether done by humans or by machines, not all properties are as easily observed and not all inferences require the same effort. In our terminology, different class structures and different choices of bases for classes might require different effort when processing information. Thus, our analysis can provide the formal grounds for introducing cost-benefit considerations into classification schemes, based on the effort that will be spent on obtaining answers by applying inferences to observed information.

Acknowledgments

This research was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada to each of the authors. The authors would like to thank attendees of the Seventh Symposium on Research on Systems Analysis and Design, as well as participants in research seminars at Ben Gurion University, Drexel University, Haifa University, Tel Aviv University, University of Canterbury, University of Minnesota, University of British Columbia, and Victoria University – Wellington, for their feedback on presentations of this research. In addition, the authors are grateful for valuable constructive feedback from the associate editor and three reviewers on earlier versions of this manuscript. Finally, the authors appreciate the feedback provided by members of the expert panel who participated in the empirical evaluation reported in the paper, and are grateful to Paul Ralph for research assistance in coding the feedback provided by the expert panel.

References

Anda, B., and Sjoberg, D. 2005. "Investigating the Role of Use Cases in the Construction of Class Diagrams," *Empirical Software Engineering* (10), pp. 285-309.

Bansiya, J., and Davis, C. 2002. "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Transactions on Software Engineering* (28:1), January, pp. 4-17.

Berners-Lee, T., Hendler, J., and Lassila, O. 2001. "The Semantic Web," *Scientific American* (284:5), pp. 34-43.

Booch, G., Rumbaugh, J., and Jacobson, I. 1999. *The Unified Modeling Language User Guide*, Reading, MA: Addison-Wesley.

Bruner, J., Goodnow, J., and Austin, G. 1956. *A Study of Thinking*, New York: Wiley.

Bunge, M. 1977. *Treatise on Basic Philosophy: Volume 3, Ontology*, Amsterdam: Reidel.

Chandrasekaran, B., Josephson, J., and Benjamins, R. 1999. "What are Ontologies, and Why Do We Need Them?," *IEEE Intelligent Systems* (14:1), January-February, pp. 20-26.

Chen, P. 1976. "The Entity–Relationship Model: Toward a Unified Model of Data," *ACM Transactions on Database Systems* (1:1), March, pp. 9-36.

Chua, C., Chiang, R., and Lim, E.-P. 2003. "Instance-Based Attribute Identification in Database Integration," *The VLDB Journal* (13:3), October, pp. 228-243.

Codd, E. 1970. "A Relational Model of Data for Large Shared Data Banks," *Communication of the ACM* (13:6), June, pp. 377-387.

Dean, J., Grove, D., and Chambers, C. 1995. "Optimization of Object-Oriented Programs Using Static Class Hierarchy Analysis." *Proceedings of the 9th European Conference on Object-Oriented Programming (ECOOP'95)*, Denmark, pp. 77-101.

Dennis, A., Haley Wixom, B., and Tegarden, D. 2002. *Systems Analysis and Design: An Object-Oriented Approach with UML*. New York: John Wiley & Sons.

Dobing, B., and Parsons, J. 2006. "How UML is Used," *Communications of the ACM* (49:4), May, pp. 109-113.

Elmasri, R., and Navathe, S. 2006. *Fundamentals of Database Systems*, Boston: Addison-Wesley.

Eriksson, H.-E., and Penker, M. 1998. *UML Toolkit*, New York: John Wiley & Sons.

Fensel, D. 2003. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Heidelberg: Springer.

Fowler, M., and Scott, K. 1997. *UML Distilled: Applying the Standard Object Modeling Language*, Reading, MA: Addison-Wesley.

Graham, I. 2001. *Object-Oriented Methods*, Reading, MA: Addison-Wesley.

Gruber, T. 1993. "A Translation Approach to Portable Ontology Specification," *Knowledge Acquisition* (5:2), pp. 199-220.

Guarino, N., and Welty, C. 2002. "Evaluating Ontological Decisions with Ontoclean," *Communications of the ACM* (45:2), February, pp. 61-65.

Hammer, M., and McLeod, D. 1981. "Database Description with SDM: A Semantic Database Model," *ACM Transactions on Database Systems* (6:3), pp. 351-386.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1), March, pp. 75-105.

Hofmann, H., and Lehner, F. 2001. "Requirements Engineering as a Success Factor in Software Projects," *IEEE Software*, 18(4), July/August, pp. 58-66.

Hull, R., and King, R. 1987. "Semantic Data Models: Survey, Applications, and Research Issues," *ACM Computing Surveys* (19:3), pp. 201-260.

Jacobson, I., Booch, G., and Rumbaugh, J. 1999. *The Unified Software Development Process*, Reading, MA: Addison-Wesley.

Kent, W. 1979. "Limitations of Record-Based Information Models," *ACM Transactions on Database Systems* (4:1), March, pp. 107-131.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind*, Chicago: University of Chicago Press.

Larman, C. 2002. *Applying UML and Patterns*, Upper Saddle River, NJ: Prentice-Hall.

Liu, D., Subramaniam, K., Far, B., and Eberlein, A. 2003. "Automating Transition from Use-Cases to Class Model," *Proceedings of the 2003 IEEE Canadian Conference on Electrical and Computer Engineering*, May, pp. 831-834.

Lieberman, H. 1986. "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems," in *ACM SIGPLAN Notices: OOPSLA'86 Conference Proceedings*, N. Meyrowitz (ed.), (21:9), September, pp. 214-223.

March, S. T., and Smith, G. 1995. "Design and Natural Science Research on Information Technology," *Decision Support Systems* (15:4), December, pp. 251-266.

Medin, D., and Smith, E. 1984. "Concepts and Concept Formation," *Annual Review of Psychology* (35), pp. 113-138.

Medin, D., Coley, J., Storms, G., and Hayes, B. 2003. "A Relevance Theory of Induction," *Psychonomic Bulletin and Review* (10:3), pp. 517-532.

Monarchi, D., and Puhr, G. 1992. "A Research Typology for Object-Oriented Analysis and Design," *Communications of the ACM* (35:9), September, pp. 35-47.

Overmyer, S., Lavoie, B., and Rambow, O. 2001. "Conceptual Modeling Through Linguistic Analysis Using LIDA," *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Canada, May, pp. 401-410.

Parsons, J. 1996. "An Information Model Based on Classification Theory," *Management Science* (42:10), October, pp. 1437-1453.

Parsons, J., and Wand, Y. 1997a. "Choosing Classes in Conceptual Modeling," *Communications of the ACM* (40:6), June, pp. 63-69.

Parsons, J., and Wand, Y. 1997b. "Using Objects in Systems Analysis," *Communications of the ACM* (40:12), December, pp. 104-110.

Rehder, B., and Burnett, R. 2005. "Feature Inference and the Causal Structure of Categories," *Cognitive Psychology* (50), pp. 264-314.

Rosch, E. 1978, "Principles of Categorization," in *Cognition and Categorization*, E. Rosch and B. Lloyd (eds.), Hillsdale, NJ: Erlbaum, pp. 27-48.

Ross, B., and Murphy, G. 1996. "Category-Based Predictions: Influence of Uncertainty and Feature Associations," *Journal of Experimental Psychology: Learning, Memory, and Cognition* (22:3), pp. 736-753.

Schmid, H., and Swenson, R. 1975. "On the Semantics of the Relational Data Model," in *Proceedings of the 1975 SIGMOD International Conference on Management of Data*, W. F. King (ed.), San Jose, CA, May, pp. 211-223.

Smith, E. 1988. "Concepts and Thoughts," in *The Psychology of Human Thought*, R. Sternberg and E. Smith (eds.), Cambridge, MA: Cambridge University Press, pp. 19-49.

Smith, E., and Medin, D. 1981. *Categories and Concepts*, Cambridge, MA: Harvard University Press.

Smith, J., and Smith, D. 1977. "Database Abstractions: Aggregation and Generalization," *ACM Transactions on Database Systems* (2:2), June, pp. 105-133.

Song, I.-Y., Yano, K., Trujillo, J., and Lujan-Mora, S. 2004. "A Taxonomic Class Modeling Methodology," in *Information Modeling Methods and Methodologies*, J. Krogstie, T. Halpin, and K. Siau (eds.), Hershey, PA: Idea Publishing Group, pp. 216-240.

Sowa, J. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Pacific Grove, CA: Brooks/Cole.

Stevens, P., and Pooley, R. 2000. *Using UML: Software Engineering with Objects and Components*, Reading, MA: Addison-Wesley.

Teorey, T., Yang, D., and Fry, J. 1986. "A Logical Design Methodology for Relational Databases Using the Extended Entity–Relationship Model," *ACM Computing Surveys* (18:2), June, pp. 197-222.

Tsichritszis, D., and Lochovsky, F. 1976. "Hierarchical Data-Base Management: A Survey," *ACM Computing Surveys* (8:1), March, pp. 105-123.

Wand, Y., and Weber, R. 2002. "Research Commentary: Information Systems and Conceptual Modeling—A Research Agenda," *Information Systems Research* (13:4), December, pp. 363-376.

Winter, S. 1998. "Evolution, Categorization and Values," Lund University Cognitive Studies 66, Lund, Sweden, pp. 1-12.

Yamauchi, T., and Markman, A. 2000. "Inference Using Categories," *Journal of Experimental Psychology: Learning, Memory, and Cognition* (26:3), pp. 776-795.

About the Authors

Jeffrey Parsons is a professor in the Faculty of Business Administration at Memorial University of Newfoundland, Canada. He received his Ph.D. from The University of British Columbia. His research interests include data management, systems analysis and design, and electronic commerce. His research has been published in journals such as *ACM Transactions on Database Systems*, *Communications of the ACM*, *IEEE Transactions on Software Engineering*, *Journal of Management Information Systems*, and *Management Science*. He is a senior editor of the *Journal of the Association for Information Systems* and serves on the editorial boards of *Information Systems Research* and the *Journal of Database Management*.

Yair Wand is CANFOR Professor of MIS at the Sauder School of Business, The University of British Columbia, Canada. He received his D.Sc. in Operations Research from The Technion (Israel) and his M.Sc. in Physics from the Weizmann Institute (Israel). His current research interests include theoretical foundations for information systems analysis and design, development and evaluation of systems analysis methods, and conceptual modeling. His published work includes articles in *ACM Transactions on Database Systems*, *Communications of the ACM*, *IEEE Transactions on Software Engineering*, *Information Systems Research*, *Journal of Information Systems*, and the *Requirements Engineering Journal*.

Appendix A

Proofs of Lemmas and Theorems

Notation

Properties and Classes

X: the set of instances. x: an instance.
P,Q,R: a set of properties. p,q,r: a property.
p(x): the properties of instance $x \in X$.
C: a set of properties defining a class for which instances exist.
Instance(C): the instances of the class defined by C.
p=v: a manifestation, v, of property $p \in p(x)$ (p=v is itself a property).
C/p=v: a subclass defined by all instances of C where p=v.
The set of properties defining C/p=v is $\{p=v\} \cup C$.
Instance(C/p=v)=$\{x \mid x \in \text{Instance}(C)$ and $(p=v) \in p(x)\}$.

Operations on Classes

Class Union: Let C_1, C_2 be classes. $C_1 \cup C_2$ is a class defined by the union of properties, assuming $\exists x \in X$, $C_k \subseteq p(x)$, k=1,2.
Implication: Instance($C_1 \cup C_2$)=Instance(C_1)$\cap$Instance(C_2).
Subclass: Let C be a class. $C' \supseteq C$ is a class defined by $C'=C \cup P$, $P \cap C=\varnothing$, $P \neq \varnothing$, assuming $\exists x \in X$, $C' \subseteq p(x)$.
Implication: Instance (C')$\subset$Instance(C).

Formalized Definitions

Definition 2 **(Precedence)**: Property p ***precedes*** property q if and only if every instance possessing property q also possesses property p (notation: $q \rightarrow p$).
Formally: $q \rightarrow p \Leftrightarrow \forall x \in X$, $q \in p(x) \Rightarrow p \in p(x)$.

Definition 3: Let Q and R be disjoint sets of properties. A ***multiple property precedence*** (alternatively, ***multiple precedence***, ***precedence***, or ***inference***) exists between Q and R if and only if every instance possessing all properties in Q also possesses all properties in R (notation: MP: $Q \rightarrow R$, or $Q \rightarrow R$).
Formally: MP: $Q \rightarrow R \Leftrightarrow \forall x \in X$, $Q \subseteq p(x) \Rightarrow R \subseteq p(x)$. Note that Q or R can be a single property.
Implication: $Q \rightarrow R \Leftrightarrow \forall\, p \in R$, $\exists Q' \subseteq Q$, $Q' \rightarrow p$

Definition 6: A set of components of $Q \rightarrow R$ will be termed a **decomposition** if and only if each property in R can be inferred from at least one component.
Formally: Let MP: $Q \rightarrow R$ and let MP_k: $Q_k \rightarrow R_k$ (k=1…n) be a set of components ($Q_k \subset Q$, $R_k \subseteq R$). $\{MP_k \mid k=1\ldots n\}$ will be a decomposition if and only if $\cup\{R_k; k=1\ldots n\}=R$.

Definition 8: A proper subset of class properties sufficient to identify an instance as a member of the class, where no property in the set is inferable from others in the set, is termed a ***base***.
Formally: Let C be a class and B, D a partition of C ($B \cup D=C$, $B,D \neq \varnothing$, $B \cap D=\varnothing$). B will be termed a ***base*** if and only if (1) $\forall p \in D$, $\forall Q \rightarrow p$, $Q \subseteq B$, and (2) no $p \in B$ exists such that $B-\{p\} \rightarrow p$.
Implication: Let C be a class, $B \subset C$, and $B \rightarrow C$. A base B* exists such that $B^* \subseteq B$.

Definition 9: A potential class C will be termed a ***proper class*** if and only if at least one (multiple) precedence $Q \rightarrow R$ exists such that $Q,R \subset C$.

Definition 11: An ***expansion*** of a class C is the union of C and another set of properties common to all class members but not part of the class definition.

Formally: Let C be a class. Assume a multiple precedence exists such that $C \rightarrow E$ and $E \cap C = \emptyset$. The class defined by $C \cup E$ is an ***expansion*** of C.

Definition 12: The ***full expansion*** of a class is the set of all properties common to all class members.
Formally: Let C be a class. Let $\{E_k \mid k=1,\ldots,n\}$ be all expansions of C. The ***full expansion*** of C is $E(C)=C \cup (\cup\{ E_k \mid k=1,\ldots,n\})$

Definition 13: A set of properties not completely included in the full expansion of the class, but sufficient to determine class membership, is termed a ***qualifying set***.
Formally: Q is a qualifying set for class C if and only if $Q \rightarrow C$, and $Q \not\subset E(C)$.

Lemmas and Theorems

Lemma 1: No decomposition exists for $Q \rightarrow R$ if and only if at least one property in R cannot be inferred from a proper subset of Q.
Proof:
(1) Assume $\exists r' \in R$ that cannot be inferred from any $Q' \subset Q$. By definition, if $Q \rightarrow R$ is decomposable, every $r \in R$ can be inferred from a $Q' \subset Q$. Hence $Q \rightarrow R$ is not decomposable.
(2) Assume $\forall r \in R$, $\exists Q_r \subset Q$ such that $Q_r \rightarrow r$, then $\{Q_r, r \in R\}$ is a decomposition. Hence, if $Q \rightarrow R$ is not decomposable, $\exists r' \in R$ that cannot be inferred from any $Q' \subset Q$.

Lemma 2: For every proper class, a base exists.
Proof: Let C be a proper class. $\exists MP: Q \rightarrow R$ where $Q,R \subset C$, $Q \cap R = \emptyset$. Let $B = Q \cup (C-R)$. By construction, $B,R \neq \emptyset$, $B \cap R = \emptyset$, and $B \cup R = C$. Since $Q \subset B$ and $Q \rightarrow R$, every property in C is either in B or inferred by properties in B. Let $B' = \{p \mid \exists B'' \subset B, B'' \rightarrow p\}$. B-B' is a base.

Lemma 3: If C is a proper class, every expansion of C is also a proper class. If B is a base of C, then B is a base of every expansion.
Proof: C is a proper class. From Lemma 2, $\exists B$, a base of C. Let the set of additional properties in the expansion be E. By the definition of expansion: $\forall x \in X$, $C \subseteq p(x) \Rightarrow E \subseteq p(x)$. Hence, $B \subseteq p(x) \Rightarrow E \subseteq p(x)$. $B \rightarrow C \cup E$ and B is a base of a class defined by $C \cup E$.

Lemma 4: A set of properties is qualifying if and only if every base can be inferred from it, and it is not a subset of the full expansion of the class.
Formally: Let C be a proper class. Q is a qualifying set if and only if $Q \rightarrow B$ exists for every base B of C and $Q \not\subset E(C)$.
Proof:
(1) Let Q be a qualifying set: By definition, $Q \subseteq p(x) \Rightarrow C \subseteq p(x)$. Since for every base, B, $B \subset C$ it follows that $B \subset p(x)$.
(2) Let B be a base and assume $Q \rightarrow B$ and $Q \not\subset E(C)$. $Q \subseteq p(x) \Rightarrow B \subseteq p(x) \Rightarrow C \subseteq p(x)$. Q is not a subset of E(C), hence $\exists q \in Q$ and $x \in$ Instance(C) such that $q \in p(x)$. Therefore, Q is a qualifying set.

Lemma 5: Every subclass of a proper class is a proper class.
Proof: Let C be a proper class and $C' \supset C$ define a subclass. C' has at least one instance. C is a proper class, hence $Q \rightarrow R$ exists in C. Since $C \subset C'$ $Q,R \subset C'$ and hence $Q \rightarrow R$ holds the subclass defined by C'. Thus, C' is a proper class.

Lemma 6: In an efficient class structure, no class possesses a base and the matching multiple precedence that is decomposable.
Proof: Let C be a class in the structure. By definition, C is a proper class. According to Lemma 2 it has a base B such that: $B \rightarrow C-B$. Assume $B \rightarrow C-B$ is decomposable. $\exists_1, B_2 \subset B$, $B_1 \cup B_2 = B$, $B_k \rightarrow D_k$ (k=1,2) and $D_1 \cup D_2 = C-B$. Define: $C_k = B_k \cup D_k$ (k=1,2). C_k (k=1,2) is a proper class, C is a subclass of C_k (k=1,2), and $C = C_1 \cup C_2$. By construction, $p \in C-B \Rightarrow p \in D_1$ or $p \in D_2$. Hence, every property preceded by B is preceded by B_1 or by B_2 and can be inferred for instances of C_1 or of C_2. Also, by construction, every property in C-B can be inferred in C_1 or from C_2. Thus, C provides no inferences beyond C_1 and C_2, in contradiction to the definition of an efficient class structure.

The following lemmas and theorems are intended to provide guidelines for class structures. We will assume the principle of maximal abstraction: every class definition includes all properties common to its instances (E(C)=C).

Support Lemma 1: Let C' be a subclass of C, $C' = P \cup C$. If C' has an inference $Q \rightarrow R$ that does not hold in C, then $Q \cap P \neq \emptyset$.
Proof: If $Q \cap P = \emptyset$ then $Q \subseteq C$, hence $Q \rightarrow R$ must hold for C. If $R \subseteq C$, $Q \rightarrow R$ holds in C. If $R \cap P \neq \emptyset$ then $C \cap P \neq \emptyset$. This is a contradiction.

Support Lemma 2: Let C' be a subclass of C, $C' = P \cup C$. For every base of C', B(C'), $B(C') \cap P \neq \emptyset$.
Proof: If C' has a base, B' where $B' \subseteq C$, then $B' \rightarrow P$ and $C \cap P \neq \emptyset$. This is a contradiction.

Support Lemma 3: Let C be a class and C'=P∪C a subclass of C where Q→R holds for C' but not for C. For C' to be non decomposable, Q must be a qualifying set for C.

Proof: If Q is not a qualifying set let C"=Q∪R∪P. C⊄Q∪R (otherwise Q→R exists in C or Q is qualifying for C). Hence, C"⊂C∪P=C' and C'=C∪C". Since Q→R holds in C" it also holds in C∪C". All inferences that hold in C' hold either in C or in C". C' is decomposable. Thus Q must be a qualifying set.

Theorem 1: A necessary condition for a subclass C' to provide a new inference with respect to a proper non-decomposable class C and be non-decomposable is that at least one of the following two conditions holds:

1. C' has at least two new properties with respect to C, and a multiple precedence: $Q_1 \rightarrow Q_2$ exists such that $P \cap Q_k \neq \emptyset$ for k=1,2.
2. C' has a subset of properties which is a qualifying set of C.

Each of the conditions is sufficient for C' to have a new inference with respect to C.

Proof

(1) It is necessary that one of the conditions holds:
Let C' be P∪C. The new inference is of the general form MP:$C_1 \cup P_1 \rightarrow C_2 \cup P_2$ where $C_1, C_2 \subseteq C$, $P_1, P_2 \subseteq P$. Based on Support Lemma 1 it must be of the form MP:$C_1 \cup P_1 \rightarrow C_2 \cup P_2$ where $P_1 \neq \emptyset$. It cannot be $P_1 \rightarrow P_2$ as then C' will be decomposable to C and P.
(1-1) $P_1, P_2 \neq \emptyset$, P has at least two elements and a precedence exists $P_1 \cup C_1 \rightarrow P_2 \cup C_2$.
(1-2) $P_2 = \emptyset$, then $C_1 \cup P_1 \rightarrow C_2$. Based on Support Lemma 3, for C' to be non decomposable $C_1 \cup P_1$ is a qualifying set for C.

(2) Each condition is sufficient to have a new inference
(2-1) $Q_1 \rightarrow Q_2$ exists such that $P \cap Q_k \neq Æ$ for k=1,2. This cannot be an inference in C.
(2-2) Let C*⊂C' be a qualifying set of C. By definition, C*→C, and C*∩P≠∅. Hence C*→C exists only for the class defined by C' but not for the class defined by C.

Immediate implication: For a subclass to have a new inference and be non-decomposable, it must either have at least two new properties or include a qualifying set for the class.

Theorem 2: Let C_1 and C_2 be proper classes where neither is a subclass of the other. Assume instances exist possessing both C_1 and C_2. A necessary condition for C=C1∪C_2 to be non-decomposable and have a new inference (with respect to those provided by C_1 and C_2) is that at least one of the following conditions holds:

1. All instances of C possess at least one property $p \in C_1 \cup C_2$.
2. C has at least one proper subset of properties that is qualifying for C_1 or for C_2.

Each of these conditions is sufficient for C to have a new inference with respect to C_1, C_2.

Note: The two conditions are not exclusive.

Proof:

(1) One of the conditions must hold:
Let Q→R be the new inference. If Q-C≠∅ or R-C≠∅ condition 1 holds. Assume this is not the case, and Q,R⊆C, then $Q=Q_1 \cup Q_2$ and $R=R_1 \cup R_2$ where $Q_k, R_k \subseteq C_k$, k=1,2.
$Q_i - Q_j \neq \emptyset$ i,j=1,2, i≠j, otherwise Q→R holds for C_1 or for C_2 and is not a new inference. If ∀p∈R, either $Q_1 \rightarrow p$ or $Q_2 \rightarrow p$, then every element in R is inferred either in C_1 or in C_2 and $Q_1 \cup Q_2 \rightarrow R$ is not a new inference. Hence at least one precedence exists Q*→p∈R where $Q^* \cap (Q_1 - Q_2) \neq \emptyset$ and $Q^* \cap (Q_2 - Q_1) \neq \emptyset$. Assume, without loss of generality, this inference is $Q^*_1 \cup Q^*_2 \rightarrow C'_1 \subseteq C_1$ where $Q^*_k \subseteq Q_k$, k=1,2. This inference does not exist in C_1, hence, according to Support Lemma 3, $Q^*_1 \cup Q^*_2$ is a qualifying set for C_1.

(2) Each condition is sufficient to have a new inference:
(2-1) Assume condition (1) holds. Let B_k be a base of C_k, k=1,2. A base B(C) exists such that $B(C) \subseteq B_1 \cup B_2$. Assume the new property is p, then B(C)→{p} is a new inference.
(2-2) Assume condition (2) holds. Let C* be a qualifying set for $C_1(C_2)$. C* must include properties of $C_2(C_1)$ which are not in $C_1(C_2)$, otherwise it is a subset of $C_1(C_2)$ and cannot be a qualifying set. Also, C* cannot be a subset of $C_2(C_1)$ as then $C_1(C_2)$ will be a subclass of $C_2(C_1)$. Hence $C^* \rightarrow C_1(C_2)$ is an inference that does not exist in C_1 or in C_2.

Immediate implication: For a class defined by the union of properties of two classes to have a new inference and be non-decomposable it must have at least one new property or include at least one subset of properties which is qualifying for at least one of the original classes.

Theorem 3: Let C be a non-decomposable proper class. Let C/p=v be a subclass of C defined by (p=v)∪C where p∈C. A necessary condition for C/p=v to have an inference in addition to those of C and be non-decomposable, is that one of the following conditions holds:

1. All instances of C/p=v possess a property (in addition to p=v) not common to all instances of C.
2. A qualifying set for C exists in C/p=v and p=v is included in the qualifying set.

Each condition is sufficient for C/p=v to have a new inference with respect to C.

Proof:

(1) One of the conditions is necessary:
Based on Theorem 1, if C/p=v has an additional inference, one of the following holds:
At least two additional properties exists in C/p=v with respect to C.
A subset of C/p=v exists such that it is qualifying for C.
- (1-1) Assume two additional properties exist in C/p=v. Since p=v is a property of all instances in C/p=v, at least one additional common property must exist in C/p=v.
- (1-2) Assume a qualifying set C* for C exists in C/p=v, and no additional class level property exists in C/p=v. $C^* \rightarrow C$ is an inference that holds for instances of C/p=v but not for those of C. According to Support Lemma 1 $(p=v) \in C^*$.

(2) Each condition is sufficient:
- (2-1) Let the new class level properties in C/p=v be P. Let $P^*=P \cup (p=v)$. According to Support Lemma 1, for every base B' of C/p=v, $P^* \cap B' \neq \emptyset$. Hence, an inference $B' \rightarrow C/p=v$ exists where $B'-C \neq \emptyset$. This is a new inference.
- (2-2) Assume $C^* \subseteq C/p=v$ is a qualifying set for C and no additional class level property exists in C/p=v. $C^* \rightarrow C$ holds in C/p=v but not in C (otherwise C* is not qualifying). Hence, this is a new inference and, according to Support Lemma 1 $C^* \cap \{p=v\} \neq \emptyset$.

Immediate implication: For a subclass defined by a value of a property to have a new inference and be non-decomposable it must either have at least one additional property, or include a qualifying set for the class where p=v is included in this set.

Appendix B

Instructions for Expert Panel Members

Task

Attached are two diagrams constituting conceptual models representing views of an application domain (*academic administration*). The diagrams are similar, but contain some differences resulting from the fact that they were constructed according to different rules for creating conceptual models.

Please examine both diagrams and answer the following:
(1) What differences can you identify between the two views?
(2) What questions might you ask to clarify the reasons for these differences?
(3) Which of the two might be more useful or realistic based on your general knowledge of the domain?

Note: We are aware these views are still partial and some information is missing. We ask that you base your responses on the diagrams as presented.

Notation:
1. A subset symbol on a line indicates a superclass/subclass relationship between two classes.
2. A circle containing an "o" (overlapping) indicates a superclass with subclasses that can have overlapping instances (an instance can belong to both subclasses).
3. A circle containing a "c" (covering or complete) indicates that the instances of a superclass must belong to at least one of its subclasses.
4. A circle containing "∪" indicates a class whose instances are a subset of the union of the instances of two other classes.

(Subsequent pages of the material contained the diagrams in Figures 1 and 2.)

Appendix C

Respondent Feedback on Treatment of SECTION and its Subclass

Resp.	Comments Favoring Original Diagram	Comments Favoring Diagram Based on Classification Rules (with reference to rules relevant to particular responses)	Questions to Ask to Clarify Domain Semantics	Which Diagram Is More Realistic?
1		In Diagram 1…it seems that the system only keeps information about which sections students have taken for current enrollment…the only connection between student and section comes through current_section. Diagram 2, on the other hand, allows a relationship directly between section and student, identifying the difference between current sections and past sections through a subset of section that contains all past sections. These past sections have a grade for each student who has participated in the section. [Rule 7]	Is it the intent of the system to keep historical records for which students have taken which sections?	With regard to the treatment of past/ current sections, clearly Diagram 2 is preferred.
2		Current_section class in diagram 1 is a superfluous class as this class is more like an instance of the class Section (with specific values, year=current and semester=current). I would suggest that this class should be removed (as done in diagram 2) and replaced with the class completed_section. [Rule 7]	When can a student get a grade (during a course or after completing a course)? Can a student get a grade without completing a course?	I think for making additional inferences of the domain, Diagram 2 is better
3			What else is there to know about the Student entity that would help understanding whether current or completed section (or both) should be captured?	
4		Diagram 2 makes more sense. Diagram 1 implies that the only sections that students can have registered with are current sections. However, in the typical university, students should have registrations associated with all sections they have taken. Moreover, as Figure 2 shows, students will have grades in completed sections. [Rule 3]		Diagram 2, because it shows sections more effectively (more in line with the typical university) and includes more information (grades).
5		In Diagram 1…there is no indication of where the grade for the section is stored. In Diagram 2 we collect information about section and completed_section. Students are registered into Section (not current sections) and completed sections have an assigned grade associated with a student. There is clarity in the differences between section and completed section. [Rules 3, 7]	Do all sections have a grade? Can a current section have a grade? What is the difference between a section and current section? What is the difference between a section and a completed section? Where is the final "grade" for a section stored?	In my experience, Diagram 2 has more clarity particularly in respect to grades. Completed sections have grades clearly agrees with my experience that grades are given at the end of a course.
6		Based on diagram 1, it seems that students can only register in current sections. Based on diagram 2, the students can register for a section (in the past, present, or the future) and receive grades for a completed section only. [Rule 2, having a grade indicates having been registered. Hence a qualifying property]	Can the students not register for future sections, e.g., in the fall semester can I not register for the spring semester of next year?	I think both would work but I like Diagram 2 a little better because it is more explicit. Also, one can register for a section, present or in the future.

Resp.	Comments Favoring Original Diagram	Comments Favoring Diagram Based on Classification Rules (with reference to rules relevant to particular responses)	Questions to Ask to Clarify Domain Semantics	Which Diagram Is More Realistic?
7		Diagram 1 is useful only as a "snapshot" of current domain state whereas, diagram 2 can also capture historical/future states that might also be of interest (should be recorded). It is still possible to hold (and retrieve) the sections a student is registered to in the present as well as those he was registered to and those he will participate-in, in the future. Furthermore, diagram 2 also specify courses a student has a grade in whereas this information is not included in diagram 1. [Rule 7]	Can you list all courses a student may have taken (been registered to) in the past? Can you identify the courses a student has a grade in? Can a student be still registered in a course which was already completed? Can you list all the courses a student is already registered for next year?	I find diagram 2 to be better.
8	Diagram 1 explicitly states what a current_ section is. Diagram 2 does not.	In diagram 2 it is clear that a student may have grades in some completed sections. Diagram 1 shows that a student may be registered in some current sections	Does a student have to be currently registered in a course in diagram 1? Does a student have to be registered (currently and/or previously) in a course in diagram 2	Diagram 2 seems more useful; it clarifies the structure of the discussed domain instance in places where it might differ from similar instances of the domain.
9		Diagram 1 – STUDENT can only be registered to a CURRENT_SECTION. This means that there cannot be any registration history stored and STUDENT has to be updated when the CURRENT_SECTIONs change. Diagram 2 – STUDENT can be registered to any SECTION: completed or not. Lacking any constraints on the model, on the one hand this allows history of STUDENT registrations to be kept. On the other hand this allows an "invalid" registration (e.g. new student registering to history).	Diagram 1 – should the model store history relating to STUDENTs registered in SECTIONS?	Diagram 2 because administration involves planning (future) and historical records. Diagram two because…it seems to make more sense to separate out that information which does not change (i.e., history in Diagram 2).
10			Can students be registered in future sections? Do students have grades in all completed sections?	D2 seems superior to D1, because it allows students to be registered in future sections of courses. This is clearly desirable. Also, it is possible in D2 to see in which completed sections students were registered (because of "has grade in"); this information is lost in D1.

Appendix D

Respondent Feedback on Treatment of INSTRUCTOR_RESEARCHER [redacted]

Resp.	Comments Favoring Original Diagram	Comments Favoring Diagram Based on Classification Rules (with reference to rules relevant to particular responses)	Questions to Ask to Clarify Domain Semantics	Which Diagram Is More Realistic
1		In Diagram two, each class has a clear set of attributes that it inherits from its super-classes. However, in Diagram One an instance of the instructor_researcher class is either a grad_student or a faculty. This means that the attributes…are not fixed. That is, each instance…has a (SN,Name, Address, Birthdate, hire_date, Salary) but then either has (office,rank,phone) or (year, degrees) depending on if it happens to be a faculty or student. In Diagram Two, there are not classes with ambiguous attri-butes. That is, all members of a class have the same attributes. [Not a potential class, therefore, not a proper class]	Are the Terms "faculty instructor" and "grad instructor" meaningful terms in the domain?	Diagram One is less complex but uses a nonstandard definition of class. Diagram Two uses standard notation but must increase complexity to achieve the same representational goal. I think I would prefer Diagram 2 generally.
2		In diagram 2, the roles of Faculty_ Instructor (FI) and Grad_Instructor (GI) are clear. Addition of FI and GI as subclasses of Instructor_Researcher (IR) makes clear who is an IR. The meaning of the class IR is not clear in diagram 1 (it is just a union of two classes). At the end, it seems to me that the additional classes –FI, GI and completed_section in diagram 2 are care-fully placed (and thus justified) because they provide additional inferences about the domain. If one cannot make such inferences then no longer these classes are necessary. [Explicit indication of the need for classes to provide inferences]	Why are the classes Faculty_ Instructor, Grad_Instructor, and completed_section necessary in diagram 2? On what condi-tions the above mentioned classes are not necessary to be modeled? How is a Faculty_Instructor different from that of a Faculty? What is the difference between Faculty_ Instructor and Grad_Instructor? [Rule 3] Who can be instructor researchers?	I think for making additional infer-ences of the domain, diagram 2 is better. In diagram 2, consider the classes FI and GI. A FI is a subset of IR and therefore can teach. A FI is a subset of faculty and therefore can conduct research (as faculty has access to grant). Again con-sider that a GI is a subclass of IR and as IR is supported by grants, therefore GI can conduct research. Again as GI is a subclass of IR and IR can teach sections, therefore GI can teach. Thus I conclude that both GI and FI can involve in research and teaching. Thus, the ability to make the additional inferences justifies the creation of the classes GI and FI.
3	Diagram 1 has the advantage of simplicity	Diagram 2 has more information		Diagram 2 has more information. The "union" notation in Diagram 1 is not easy to understand.
4			I would want to know whether there was anything special that differentiated these two types of instructor. [Rule 3]	This seems to be a distinction without a difference.

Resp.	Comments Favoring Original Diagram	Comments Favoring Diagram Based on Classification Rules (with reference to rules relevant to particular responses)	Questions to Ask to Clarify Domain Semantics	Which Diagram Is More Realistic
5		In Diagram 1 …it is not clear that a instructor_reseracher can be both a faculty member and a grad student. It is clear that instructor researchers must be either faculty or a grad_student. In Diagram 2 the "Instructor_Researcher" entity is split into two subtypes (Faculty Instructor or Grad Instructor) where the condition is that an instructor-researcher must be at least one of these subtypes and possibly both.	Can a graduate student be a faculty member (with office, rank and phone)? Can a faculty member be a graduate student? Can a faculty member be a teaching assistant? Are all graduate students employees? Do all instructor_ researchers have an office, rank and phone number? [Not a potential class, therefore, not a proper class]	Diagram 2 allows for differences between grad instructors and faculty instructors and suggests that a person could be both a faculty member and a graduate student. This describes my last year in my PhD program.
6	In diagram 1, the instances of INSTRUCTOR_ RESEARCHER are a subset of FACULTY and GRAD_ STUDENT. In diagram 2, the subset of the classes FACULTY and GRAD_STUDENT are shown explicitly, i.e., FACULTY_Instructor and GRAD_Instructor, which in turn are "covered" by INSTRUCTOR_RESEARCHER. However, I think the two notations essentially mean the same.		Why is FACULTY_Instructor and GRAD_Instructor shown explicitly in diagram 2? Is it because in the future, there may some relationships from FACULTY_Instructor and GRAD_Instructor to other classes? [Rule 5]	I think both would work but I like diagram 2 a little better because it is more explicit.
7	The two subclasses (instructor-faculty and instructor researcher) and the completeness constraint make Diagram 2 equivalent with respect to instructor related constraints to Diagram 1.		Why do you need them [the additional subclasses]? [Rule 3]	D1, as a developer, would follow the principle of "short and simple," meaning "only show what is needed"
8		In diagram 2 I know that instructor_researcher is either a grad_instructor or a faculty_instructor. It seems that typically not all grad_students are instructor_researcher's and that not all faculty are instructor_researchers. In diagram 1 it is not clear if all faculty/ grad students are instructor_researchers.	are all grad_students also instructor_researchers in diagram 1. Are all faculty also instructor_Researchers in diagram 1.	Diagram 2 seems more useful; it clarifies the structure of the discussed domain instance in places where it might differ from similar instances of the domain. In other words… providing info on what instructor_researcher means, and whether it includes all students or all faculty seems more beneficial.
9		INSTRUCTOR_RESEARCHER has neither FACULTY nor GRAD_STUDENT attributes. [Not a potential class, therefore, not a proper class]	What attributes must/may an INSTRUCTOR_RESEARCHER have? [Not a potential class, therefore, not a proper class]	Diagram 2 because it somewhat separates INSTRUCTOR from RESEARCHER
10		D2 is a little more explicit about the fact that only a subset of the faculty and of the graduate students are instructors; however, this is only a difference in presentation, not in content. [In D1 instructor_researcher was not a proper class. In D2 all classes are proper classes]		