The Knowledge Level Approach
To Intelligent Information System Design

Michael A Zang
Senior Software Engineer
CDM Technologies, Inc
San Luis Obispo, CA, USA
mzang@cdmtech.com

Abstract

Traditional approaches to building intelligent information systems employ an ontology to define a representational structure for the data and information of interest within the target domain of the system. At runtime, the ontology provides a constrained template for the creation of the individual objects and relationships that together define the state of the system at a given point in time. The ontology also provides a vocabulary for expressing domain knowledge typically in the form of rules (declarative knowledge) or methods (procedural knowledge). The system utilizes the encoded knowledge, often in conjunction user input, to progress the state of the system towards the specific goals indicated by the users. While this approach has been very successful, it has some drawbacks. Regardless of the implementation paradigm the knowledge is essentially buried in the code and therefore inaccessible to most domain experts. The knowledge also tends to be very domain specific and is not extensible at runtime. This paper describes a variation on the traditional approach that employs an explicit knowledge level within the ontology to mitigate the identified drawbacks.

Keywords

Data, Information, Knowledge, Knowledge Management Ontology, Object Model, UML

Introduction

This paper employs a simple example to describe the knowledge level approach employed in several of the software projects currently being developed at CDM Technologies, Inc. CDM Technologies specializes in the development of collaborative decision support systems for large government and private organizations particularly in the field of maritime logistics. The example builds a simple medical diagnostic model and accompanying agent rules capable of diagnosing infection types and of recommending actions to assist in the diagnosis. The model and rules are first developed using what this paper calls the traditional approach. Next, an interim technique, termed the taxonomic approach, is developed to address some of the shortcomings identified in the traditional approach. Then the knowledge level approach is developed to address some of the shortcomings identified in the taxonomic approach. Finally, summarizing conclusions are provided, which identify the strengths and weaknesses of the knowledge level approach and provided guidance as to when it should be considered for use.

The progression from the traditional approach to the taxonomic approach to the knowledge level approach parallels those taken by the ARES development team at CDM Technologies in the successive development of three projects sponsored by the United States Office of Naval Research (ONR). These systems are: the Collaborative Agent Based Control and Help System (COACH), the Ordnance Tracking and Information System (OTIS), and the Shipboard Integration of Logistics Systems Mission Readiness Assessment Tool (SILS MRAT). This effort extensively leverages the work of Martin Fowler described in his book Analysis Patterns, Reusable Object Models (Fowler 1997a) and the work of David Hay described in his book Data Model Patterns, Conventions of Thought (Hay 1996).

This paper assumes but does not require a rudimentary knowledge of the basic concepts of object-oriented modeling. A good introduction to this subject can be found in the book Inside the Object Model by David Papurt (Papurt 1995). All the figures in this paper use a small subset of the graphical object-oriented notations defined by the Unified Modeling Language (UML). A brief overview of the UML notations employed in this paper is provided in Figure 1. A concise summary of UML can be found in UML Distilled by Martin Fowler (Fowler 1997b). The UML based figures in this document provide only the minimum level of detail necessary to understand the concepts under discussion, and therefore they leave off many of the details typical in UML diagrams such as role names and multiplicity constraints. This paper capitalizes and italicizes ontological class names, quotes and italicizes object instance names, and italicizes association, attribute and method names. Class, attribute, and method names are word separated by underscores while association names are word separated by dashes.
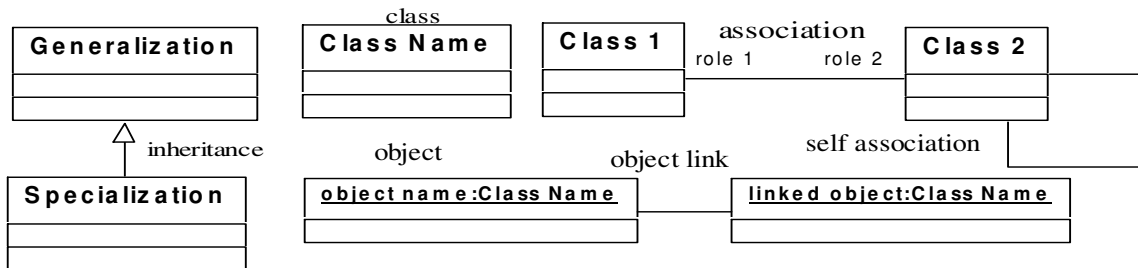
Figure 1: UML Notions Employed in this Paper

Traditional Approach

The traditional approach utilizes a statically compiled ontology that virtually mirrors the real-world entities associated with the targeted system domain. Ontology development is followed by developing agent rule sets, which are grounded in the vocabulary and structure the ontology provides, to produce the desired intelligent behavior. Following this approach an ontology for the simple medical diagnostic domain must first be developed.
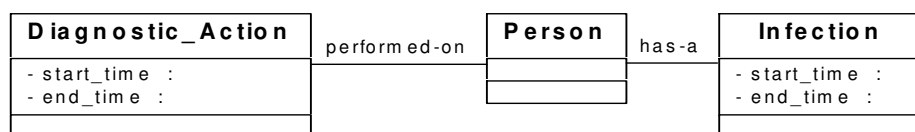
Figure 2: Mirror Image Ontological Framework

At the highest level of abstraction, the example ontology consists of three entities: *Person*, *Infection*, and *Diagnostic_Action*. Both *Diagnostic_Action* and *Infection* are temporal and therefore contain attributes to indicate the applicable time span. These entities are related in that a *Person* may optionally have an (*has-a* association) *Infection* and a *Diagnostic_Action* is *performed-on* a *Person*. This level does not provide enough detail for a diagnostic agent to perform any useful tasks but does provide the structural framework, depicted in Figure 2, with which to further develop the ontology. In order to make this a bit more interesting the diagnostic agent needs to be provided with some different types of *Infection* to diagnose. In this regard, The *Infection* class can be further specialized into *Bacterial_Infection* and *Viral_Infection* as shown in Figure 3. *Person* can also be specialized into two types: *Young_Person*, and *Old_Person*. These additions are shown in Figure 4.
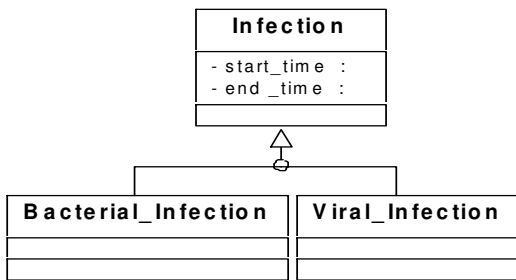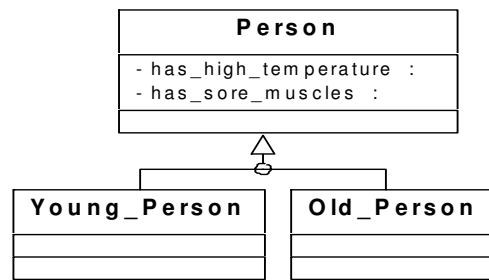

Figure 3: Types of Infection

Figure 4: Types of Person

For the sake of simplicity, assume that bacterial infections are indicated by a high fever and viral infections by sore muscles. In this regard at least two types of *Diagnostic_Action* are required: *Body_Temperature_Measurement* and *Sore_Muscle_Check*. To make things more interesting, *Body_Temperature_Measurement* can be further specialized into *Oral_ Temperature_Measurement* and *Aural_Temperature_Measurement* as shown in Figure 5. It will be assumed that the *Diagnostic_Action Oral_Body_Temp_Measurement* applies only to an *Old_Person* while *Aural_Body_Temp _Measurement* applies only to a *Young_Person*. A place is needed to record the results of these diagnostic actions. For this purpose an attribute *has_high_temperature* and an attribute *has_sore_muscles* (both true or false) can be added to the *Person* class as shown in Figure 4.
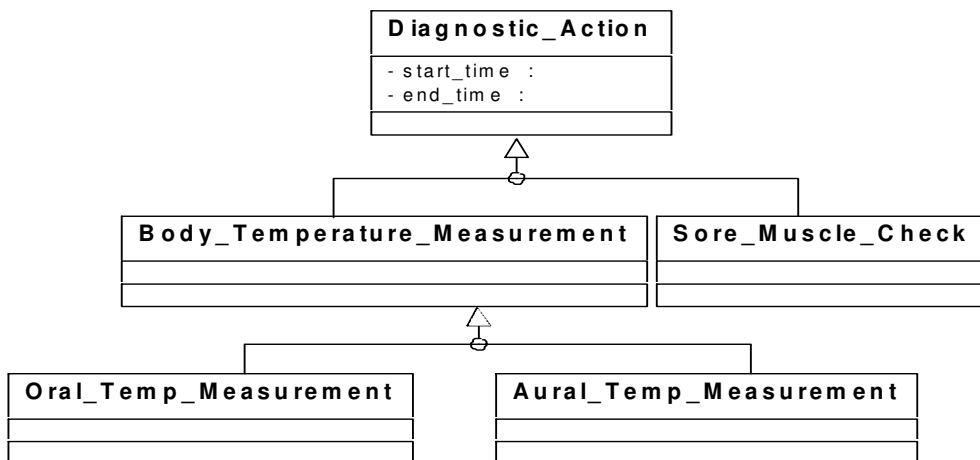

Figure 5: Types of Diagnostic Action

This completes the traditional approach developed ontology for the simple medical diagnostic example. Note that while the ontology was developed with the intended usage in mind it does not capture the associated agent rules in any manner. These may be specified in a declarative manner using condition action pairs as listed in Table 1. The rule conditions specify patterns of linked objects and are therefore specified in terms of the class names that the ontology defines. Since the diagnostic agent is targeted to diagnose types of infection, it should not be triggered until a person is known to have an undiagnosed infection. In terms of the ontology, an undiagnosed infection is indicated by the association of an object that is a kind of person (instance of class *Person*, or of a subclass of class *Person*, ad infinitum) to an instance of class *Infection* (not *Viral_Infection* or *Bacterial_Infection*). The rule scheme employs a priority to control the order in which triggered actions will be invoked.

Table 1: Diagnostic Agent Rules for the Traditional Approach

	Condition	Action	Priority
1	A kind of *Person* has_sore_muscles	Indicate *Person has-a Viral_Infection*	1
2	A kind of *Person* has_high_temperature	Indicate *Person has-a Bacterial_Infection*	1
3	A kind of *Old_Person has-a* undiagnosed *Infection*	Recommend *Oral_Temp_Measurement* performed on *Person*	2
4	A kind of *Young_Person has-a* undiagnosed *Infection*	Recommend *Aural_Temp_Measurement* performed on Person	2
5	A kind of *Person* has-a undiagnosed *Infection*	Recommend *Sore_Muscle_Check* performed on *Person*	3

The core strengths of the traditional approach are that the resulting ontologies are typically easier to understand, particularly for the uninitiated, than other approaches and typically results in more efficient implementations of agent behavior as modern languages natively support operations associated with the mirror image type of classifications hierarchies upon which a large percentage of agent logic is typically based.

A primary drawback of the traditional approach is that the agent logic dependent classification hierarchies are not easily modifiable at runtime because the class model must be extended which in turn requires recompilation. In addition, the traditional approach tends to produce models that are not reusable in the context of other domains. Since the agent and application logic of a typical information system are built directly on top of the ontology, these too will find little reuse in the context of different domains. Finally, the traditional approach does not readily support the common real-world concepts of dynamic and multiple classifications that are introduced in conjunction with the taxonomic approach in the following section.

Taxonomic Approach

The taxonomic approach utilizes a statically compiled ontology that is more abstract and generic than that employed by the traditional approach, but can be tailored to a particular domain using runtime instances that capture the specialized or unique concepts within it. In this approach, the logical classification of an object is provided by associative mechanisms

rather than the native classification mechanisms provided by the implementation language, which is employed only for the purpose of inheritance mechanisms it provides to gather up the attributes, associations, and behaviors of a particular class of object.

With the taxonomic approach, the classes of the statically compiled model are partitioned into two distinct categories: *Operational_Object* and *Taxonomic_Object* as shown in Figure 6 for the simple medical diagnostic example. The *Operational_Object* classes: *Action*, *Asset*, and *Observation* can be respectively substituted for the classes: *Diagnostic_Action*, *Person*, and *Infection*, the difference being that the logical classification of instantiated objects, upon which much reasoning by intelligent software agents can be applied, is provided by specific associations to subtypes of the *Taxonomic_Object* class. Note that concepts of action, asset, and observation from the taxonomic approach are much more general than the traditional approach concepts of diagnostic action, person, and infection and are therefore applicable to a much broader domain than that of the medical diagnostic example.
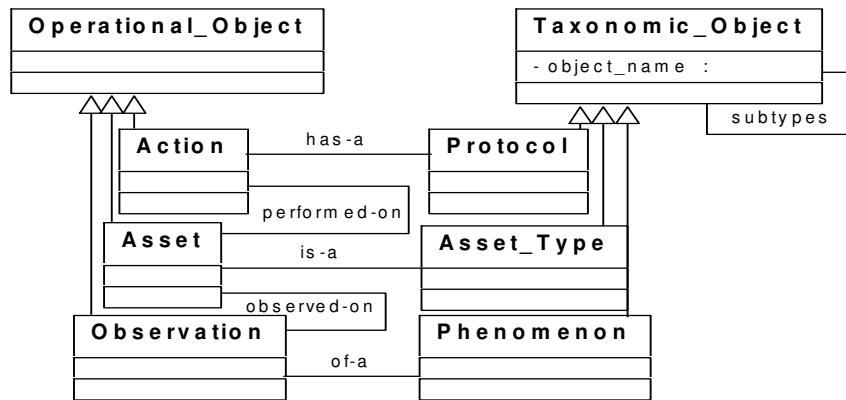

Figure 6: Taxonomic Class Model

A key part of the taxonomic approach ontology is the subtypes association of the *Taxonomic_Object* class. This allows object instances created from the *Taxonomic_Object* class to be linked together to form taxonomies that can be iterated over at runtime to provide a much more flexible classification scheme than that provided by the traditional approach. The taxonomies that substitute for the classification provided by class hierarchy of the traditional approach are shown in Figure 7 for the simple medical diagnostic example. One can easily see the *Infection* (Figure 3), *Person* (Figure 4), and *Diagnostic_Action* (Figure 5) classification hierarchies mirrored in the structures of linked object instances of the respective *Protocol*, *Asset_Type*, and *Phenomenon* classes from the taxonomic approach.
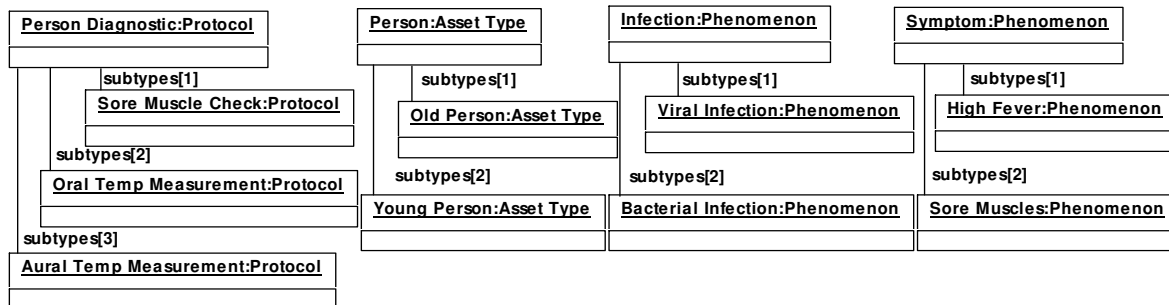

Figure 7: Taxonomic Approach Taxonomies

5

A *Phenomenon* hierarchy for symptoms can be defined so that observations of symptomatic phenomenon on '*Person'* Assets can be used to eliminate the need for the *has_sore_muscles* and *has_high_fever* attributes required for objects of class *Person* from the traditional approach ontology (Figure 4). This pattern of posting observations on phenomenon to replace attributes of the *Asset* class eliminates the need for complex inheritance hierarchies that traditionally tie attributes to classes making a domain neutral statically compiled ontology a feasible system design and development option.
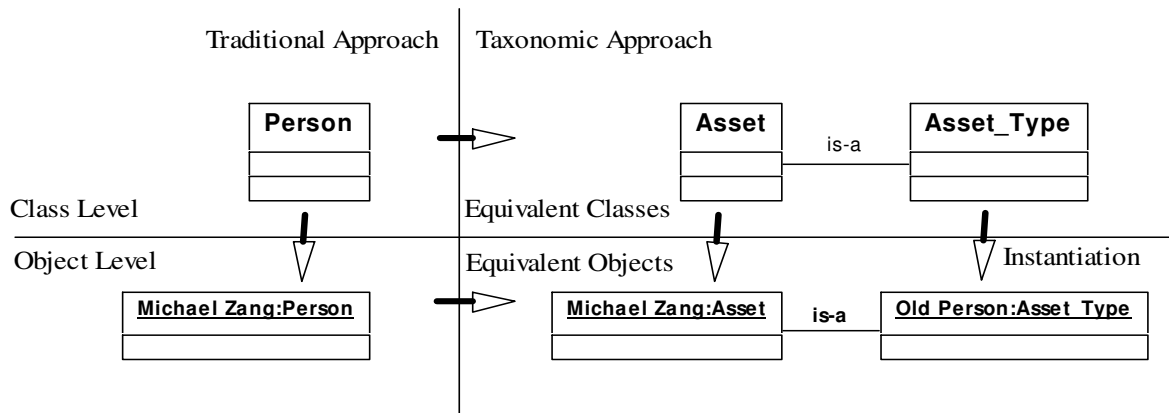

Figure 8: Equivalent Representations of Person

In order to provide the same logical meaning as objects from the traditional approach, objects instantiated from *Operational_Object* classes must be associated with an object instantiated from the corresponding *Taxonomic_Object* class. In this manner, an object instantiated from the *Person* class of the traditional approach is logically equivalent to an object instantiated from the *Asset* class of the taxonomic approach and associated to an object instance of the *Asset_Type* class with an *object_name* attribute value of '*Person'* as shown in Figure 8.
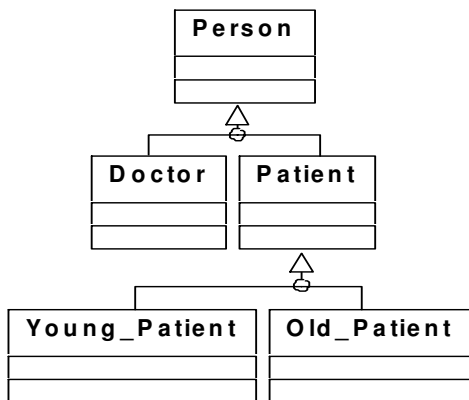

Figure 9: Extended Person Class Hierarchy   **Figure 10: Extended Person Taxonomy**
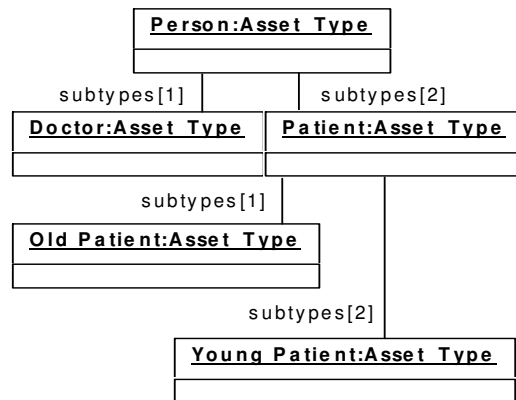
In addition to providing support for extensibility at runtime, the taxonomic approach also supports the concepts of dynamic and multiple classification both of which are common in practice but difficult to implement using the traditional approach. Dynamic classification refers to the ability of an object to change its classification at runtime. Multiple classification refers to the ability of an object to belong to more than one class. The ongoing medical

6

diagnostic example has been extended in Figure 9 for the *Person* class hierarchy of the traditional approach and in Figure 10 for the *'Person'* taxonomy of the taxonomic approach in order to provide examples of these concepts.

The example extension indicates diagnostic actions are *performed-on* a *Patient* and *performed-by* a *Doctor*. This is shown in Figure 11 for the traditional approach and in Figure 12 for the taxonomic approach. These extensions show that the flexibility provided by the taxonomic approach in regards to classification and runtime modification comes at the cost of additional complexity. This is evidenced by the complex constraint on the *Action* class that is required to, for example, prevent patients from diagnosing themselves.

Suppose a doctor gets sick and needs to be admitted to a hospital as a patient. With the taxonomic approach, this situation is represented by breaking the link between the representative *Asset* object and the *Asset_Type* object with *object_name 'Doctor'* and connecting it instead to the *Asset_Type* object with *object_name 'Patient'*. With the traditional approach this situation is much more difficult to deal with because the representative object and its classification are inseparable. The representative object of class *Doctor* must be destroyed and a new object of class *Patient* created. This process results in a loss of identity, which, in turn, results in a complete loss of the professional history (i.e. diagnostic actions performed on patients) of the doctor as the traditional approach physically constrains *Patient* objects from linking to *Diagnostic_Action* objects with the *performed-by* association. Although the taxonomic approach preserves the individual identity of the *Asset* object as the logical classification dynamically switches from *'Doctor'* to *'Patient'*, there is still an issue with the logical constraint put in place to mimic the physical constraints inherent in the traditional approach. While the logical constraint could be relaxed to deal with this, a better approach is to employ multiple classification.
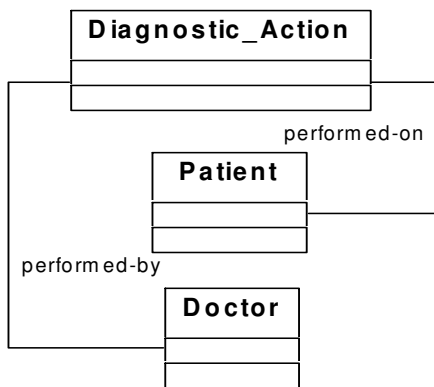

Figure 11: Extensions for Traditional Approach   **Figure 12: Extensions for Taxonomic Approach**
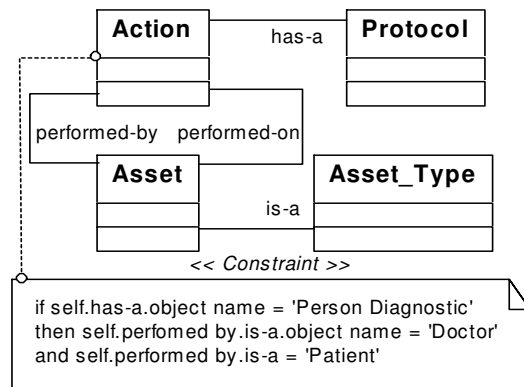
Multiple classification allows the person in question to be both a doctor and a patient, thus preserving both identity and history. This is easily accomplished using the taxonomic approach by changing the multiplicity of the *is-a* association between the *Asset* and *Asset_Type* classes from exactly one to one or more. This allows multiple *Asset_Type* instances to be associated with an *Asset* instance; thereby, allowing the *Asset* instance of the example to be logically classified as both a *'Doctor'* and a *'Patient'*.

The concept of multiple classification is difficult to implement using the traditional approach, which combines the concepts of inheritance and classification. In order to create objects that are classified as both a *Patient* and a *Doctor* in the traditional approach, language provided multiple inheritance mechanisms must be used to create a new class *Doctor_Patient* that inherits from both the *Doctor* class and the *Patient* class (Figure 13). While this in itself is messy, additional complications are incurred because the diagnostic agent rules (specified in Table 1) require that a patient be additionally classified as young or old; thereby, requiring additional usage of multiple inheritance to create classes *Young_Doctor_Patient* and *Old_Doctor_Patient*. This approach dilutes the clarity of the classification hierarchy and quickly becomes untenable in realistically scoped models.
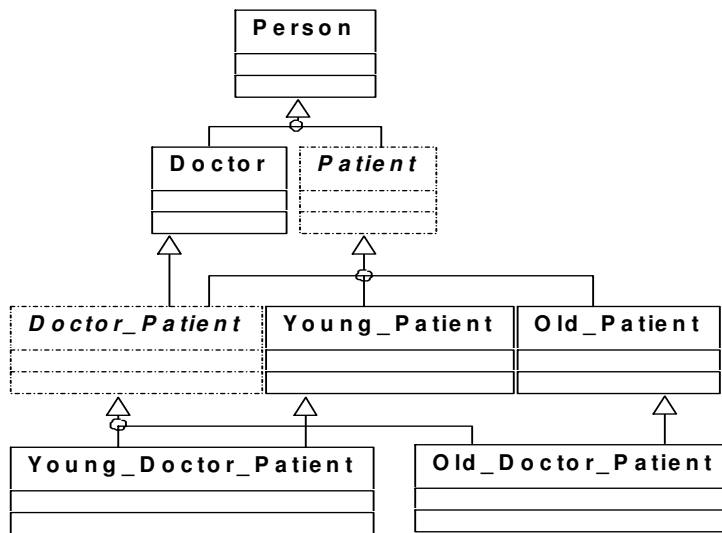

Figure 13: Multiple Classification Problems with the Traditional Approach

The taxonomic approach results in rules with more complex conditions than those resulting from the traditional approach. The specified condition for rule number 1: "*Observation* of '*Sore Muscles*' on *Asset* that is a kind of '*Person*'" is shorthand. A more rigorous specification is "an *Observation* object linked to a *Phenomenon* object, through the *of-a* association between the *Observation* and *Phenomenon* classes, of type '*Sore Muscles*', that is also linked to an *Asset* object, through the observed-on association between the Observation and *Asset* classes, that is a kind of '*Person*'". Further, note that "of type '*Sore Muscles*'" is shorthand for "a *Phenomenon* object that has an *object_name* attribute with value equal to the character string '*Sore Muscles*'. Also, note that "is a kind of '*Person*'" is shorthand for an *Asset* object linked to an *Asset_Type* object, through the *is-a* association between the *Asset* and *Asset_Type* classes, that has an *object_name* attribute with value equal to the character string '*Person*' or that has parent *Asset_Type* objects in the taxonomic tree formed by the subtypes association defined for the *Asset_Type* class. Additional complexity is required for rule condition specification in the presence of multiple classification as set notation is then required.

The complexity in rule specification can be alleviated some by providing convenience methods within the *Operational_Object* classes that mimic the native language provided behavior that was abandoned in the taxonomic approach to separate identity and inheritance

from classification. Considering the more rigorous example specification of the previous paragraph, a method named *of_type* that takes a character string as an argument and returns true or false can be added to the *Observation* class that walks *of-a* links to associated *Phenomenon* objects and compares the values of their *object_name* attributes to the string passed in as an argument. A similar method named *kind_of* can be added to the *Asset* class to walk links to associated *Asset_Type* objects then recursively searches up the taxonomic tree looking for objects with *object_name* attribute values equal to the string passed in as an argument. This sort of model dependent and domain independent behavior is ideal for implementation by statically compiled class methods.

Table 2: Diagnostic Agent Rules for the Taxonomic Approach

	Condition	Action	Priority
1	*Observation of_type 'Sore Muscles' observed-on Asset that is a kind of 'Person'*	*Observation of_type 'Viral Infection' observed-on Person*	1
2	*Observation of 'High Fever' observed-on Asset that is a kind_of 'Person'*	*Observation of_type 'Bacterial Infection' observed-on Person*	1
3	*Observation of 'Infection' on Asset that is a kind_of 'Person'*	Recommend *Action of_type 'Sore Muscle Check' be performed-on Person*	2
4	*Observation of_type 'Infection' on Asset that is a kind_of 'Young_Person'*	Recommend *Action of_type 'Aural Temp Measurement' be performed-on Person*	2
5	*Observation of_type 'Infection' on Asset that is-a kind_of 'Old_Person'*	Recommend *Action of_type 'Oral Temp Measurement' be performed-on Person*	3

The taxonomic approach appears to have addressed many of the shortcomings identified with intelligent information systems developed using the traditional approach. The abstract statically compiled ontology of the taxonomic approach is generally applicable to any collaborative, intelligent agent based (human and software) information system. The taxonomic level of the model serves as a constraining meta model that can be extended and specialized for a specific target domain by instantiating objects from the meta-level classes and configuring them to be representative of the concepts within a domain by linking them together into runtime navigable taxonomies. This flexibility comes at the cost of additional complexity, as it requires the logical classification provided by the ontology be represented using an associative pattern rather than the mechanisms provided directly by the implementation environment. In addition to providing for runtime extensibility of the core ontology, the associative classification pattern allows for a richer and a more dynamic information environment by seamlessly supporting the fundamental concepts of dynamic and multiple classification.

The domain neutral, statically compiled ontology naturally leads to powerful domain neutral application components such as observation recorders, action schedulers, and taxonomy builders. Rather than hard coding such things as selection menu choices and graphical

display layouts, system applications query the ontological model at runtime to configure themselves appropriately for both the target domain and the current user. This sort of dynamic querying is very applicable to the highly optimized, statically compiled, procedural (albeit event driven and object-oriented) environments commonly employed in the development of highly interactive applications and interfaces. Unfortunately, it is not as well suited for the declarative rule based environments commonly employed in development of intelligent agents intended to assist users in making sense of and utilizing the information and knowledge stored within the underlying software system. This is evident in the rule condition specifications for the taxonomic approach. Notice that the rule conditions in Table 2 specify patterns that include not only the statically compiled class names employed in the specification of rule conditions in the traditional approach (Table 1) but the textual values of linked object instance names as well.

The taxonomic approach successfully addresses all the issues identified with the traditional approach except the need for domain independent agent logic. When applying the taxonomic approach, one starts with an abstract, domain independent, ontology and powerful, domain neutral, application tools. Then the specialized taxonomies applicable to the domain are created from object instances of the *Taxonomic_Object* classes defined by the ontology, perhaps with the assistance of domain neutral application tools designed for the construction and maintenance of these sorts of domain specific ontologies. Finally, agent logic, based on both the statically compiled ontology and the specialized linked object taxonomic structures for the domain, is develop to provide intelligent collaborative support for system users. While it is possible to extend this agent logic at runtime as most declarative rule based inference engines support the dynamic loading and interpretations of rules at runtime, the corresponding rule development environments have not typically been accessible to even the most advanced users of typical information systems, which greatly compromises the user extensibility of the taxonomic approach.

Note however, that recent advances in applied artificial intelligence are beginning to result in reasoning facilities with that are more accessible to technically savvy subject matter experts or applicable to supervised or unsupervised algorithmic learning approaches. An example of such is the Taxonomic Case-Based Reasoning System (TCRS) (Aha 2002)(Gupta 2001) that has been successfully utilized in the development of CDM systems employing the taxonomic approach. TCRS is particularly well suited to the taxonomic approach, and by extension the as yet to be introduced knowledge level approach, because it employs taxonomically linked objects to tailor the characteristic question and answer dialogs associated with case retrieval to the level of expertise of the user.

Knowledge Level Approach

The knowledge level approach addresses the single identified shortcoming of the taxonomic approach by further extending the fundamental tenets of the approach by inter linking the taxonomic object instances, through logically typed associations, to record additional knowledge about them and the associated usage of them by the objects in the operational level. Unlike the rule-encoded knowledge employed the traditional and taxonomic approaches, the knowledge recorded through logically typed associations is in a form that is both dynamically extensible and conceptually accessible by system users. The ontology

developed for the simple medical diagnostic example using the knowledge level approach is depicted in Figure 14. It can be readily seen that basic elements and structure of the ontology are the same as in the taxonomic approach except for two significant differences: the generalization of all linkages between levels and the additional associations defined within levels.

In order to both formalize and standardize the use of associations to knowledge level classes to provide logical classification to instances of operational level classes a single *type-of* association between the *Operational_Object* class and the *Knowledge_Object* class has been provided. This association substitutes for the individual associations defined between the *Action* and *Protocol*, *Asset* and *Asset_Type* and *Observation* and *Phenomenon* classes in the taxonomic approach (Figure 6). The generalization of these associations allows generic implementations of the type_of and kind_of convenience methods to be applicable to all subtypes of the *Operation_Object* class. This generalization requires the addition of fixed constraints on the *Action*, *Asset*, and *Observation* classes.
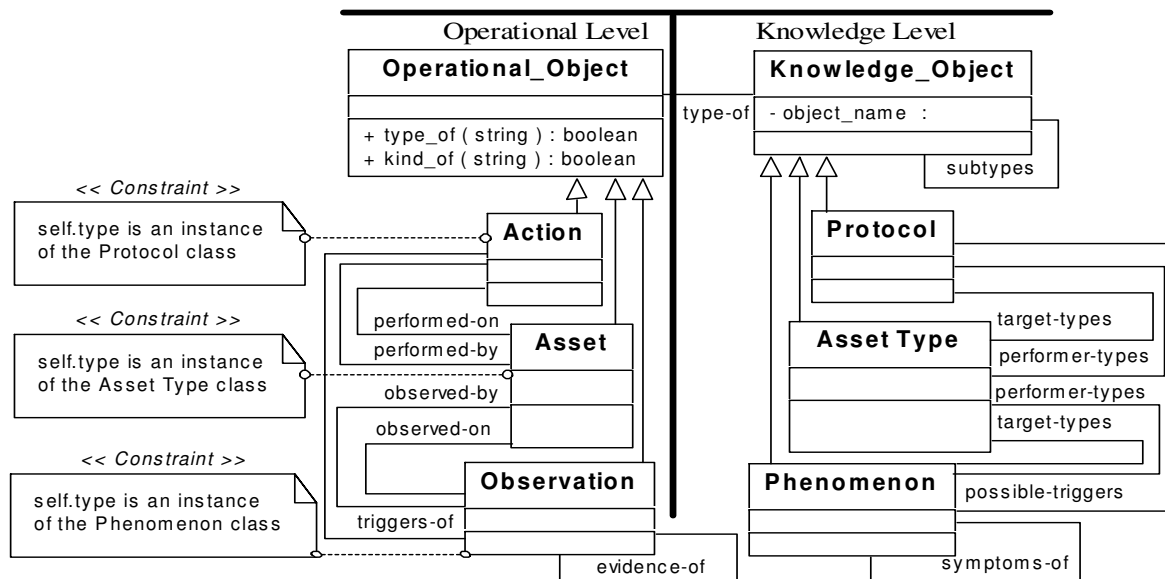

Figure 14: Knowledge Level Approach Ontology

In order to eliminate the agent logic dependence on specific object instances in the taxonomies formed through the subtypes association defined for *Knowledge_Object* classes exhibited by the rules developed using the taxonomic approach (Table 2) the self-association symptoms-of has been added to the Phenomenon class and the association possible-triggers has been added between the Phenomenon and Protocol classes. Set membership in the object links formed by these associations is used as a substitute for the hard coded *object_name* attribute values required by the taxonomic approach rules. This allows for the creation of domain independent rules based only on the generic statically compiled ontology and set operations.

The symptoms-of association allows a single domain independent rule (rule 1 in Table 3) to replace the two domain specific diagnostic observation rules developed using the traditional

and taxonomic approaches (rules 1 and 2 in Table 1 and Table 2). The possible-triggers association allows a single domain independent rule (rule 2 in Table 3) to replace the three domain specific diagnostic action recommendation rules developed using the traditional and taxonomic approaches (rules 3, 4, and 5 in Table 1 and Table 2). By cross-linking the taxonomic concept hierarchies using logical associations the essence of the rules developed under the traditional and taxonomic approaches has been moved into the form of instance data that can be readily extended at runtime just as the taxonomic approach allowed for runtime extensions of the core concepts within the ontology.

Table 3: Diagnostic Agent Rules for the Knowledge Level Approach

	Condition	Action	Priority
1	*Observation* of *type_of Phenomenon observed-on Asset* with *type_of Asset_Type* in *Phenomenon target-types* with parent *symptoms-of* link	Create *Observation* instance observed-on Asset of type_of Phenomenon equal to the Phenomenon associated as a parent with the symptoms-of link	1
2	*Observation* on *type-of Phenomenon observed-on Asset* with *type_of Asset_Type* in *Phenomenon target-types* and a *Protocol* in *possible-triggers*	Recommend Action of-type *Protocol* be performed-on *Asset*	2

The rules that remain under the knowledge level approach act as domain generic machinery for reasoning on the domain specific knowledge instance models. The domain specific knowledge instance models (interlinked *Knowledge_Object* instances) are loaded at runtime or created by advanced users to tailor the statically compiled, domain independent ontology to support the specialized concepts with in the target system domain. By adding new linkages, which exist as data elements rather than code, an unlimited number of rules like those developed under the traditional and taxonomic approaches can be added to the system at runtime. These new linkages can just as easily be connected to new user added concepts as to existing ones; thereby, eliminating the problem identified for the taxonomic approach.

Summary

The knowledge level approach to developing intelligent information systems utilizes an abstract, domain independent, statically compiled ontology divided into two distinct levels. The operational level provides classes to serve as templates for creating object instances that record the day-to-day events within the domain. The knowledge level provides classes to serve as templates for creating object instances to record domain specific concepts and knowledge of their application. Rather than using the language provided classification mechanisms operational level objects associate with knowledge level object to represent information related to their logical classification. This approach provides support for the powerful modeling concepts of dynamic and multiple classification and allows for the development of generic statically compiled ontologies that can be reused across multiple disparate domains.

The statically compiled knowledge level provides a control structure and generic rule activation mechanisms that system developers, subject matter experts, or advanced users may utilize to tailoring the generic ontology to address the specialized or unique concepts within a particular system domain. The fixed statically compiled ontology also allows for the development of powerful, domain neutral, application tools such as: action schedulers, observation recorders, and taxonomy editors that leverage the knowledge recorded by knowledge level instances in order to tailor the application and its interface to the specialized requirements of the domain. Ultimately the knowledge level approach is a structural layering pattern used in the specification of ontologies for intelligent information systems. A well-designed ontology may be layered in other compatible dimensions as well and examples of this are provided in (Pohl 2000) and (Zang 2002).

The knowledge level approach is not necessarily applicable to development of all information system. Although it reduces complexity by reducing both the number of classes and the number of rules, it increases complexity in other ways that make ontologies developed using the knowledge level approach much more difficult to understand for novice programmers and for experienced programmers new to a knowledge level approached based project. The knowledge level approach is particularly applicable for use by development teams involved in the development of multiple (concurrent or over time) information systems that have focus on either intelligent agents or knowledge management.

References

Aha, David W. and Gupta, Kalyan Moy (2002), Causal Query Elaboration in Conversational Case-Based Reasoning; Proceedings of FLAIRS'02

Fowler, Martin (1997a); Analysis Patterns, Reusable Object Models; Addison Wesley Longman

Fowler, Martin and Kendall Scott (1997b); UML Distilled, Applying the Standard Object Modeling Language; Addison Wesley Longman

Gupta, Kalyan Moy (2001); Taxonomic Conversational Case-Based Reasoning; Proceedings of the fourth ICCBR, D. W. Aha & I. Watson (Eds.), Springer, Berlin, Germany, pp. 219-233

Hay, David C. (1996); Data Model Patterns, Conventions of Thought; Dorset House

Papurt, David M (1995); Inside the Object Model; Sigs Books

Pohl, Kym (2000); Perspective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems; Collaborative Agent Design Research Center, Cal Poly San Luis Obispo, Ca

Zang, Michael A (2002); Data, Information, and Knowledge in the Context of SILS; Proceedings ONR Workshop Series on Collaborative Decision-Support Systems; Office of Naval Research Logistics Program Office