Data, Information, and Knowledge in the Context of SILS

Michael A Zang
Senior Software Engineer
CAD Research Center
Cal Poly, San Luis Obispo, CA 93407
mzang@cadrc.calpoly.edu

Abstract

Data, information, and knowledge are becoming increasingly common terms in the literature of the software industry. This terminology originated some time ago in the disciplines of cognitive science and artificial intelligence to reference three closely related but distinct concepts. Traditionally, mainstream software engineering has lumped all three concepts together as data and has only recently begun to distinguish between them. Unfortunately, the popular desire to distinguish between data, information, and knowledge within the mainstream has blurred the individual meanings of the words to the point where there is no longer a clear-cut distinction between them for most people. This problem is compounded by the fact that the abstract nature of the associated concepts provides wide latitude for their application.

The goal of this paper is to make these abstract concepts more concrete by providing examples of their usage taken directly from the design and implementation of the Shipboard Integration of Logistics Systems (SILS), an ONR project sponsored by Dr. Phillip Abraham. This paper does not claim or intend to provide definitive definitions of these terms; rather it seeks to provide a cognitive framework for thinking about these concepts from which observations and conclusions can be made about the differences and relationships between the individual concepts.

Keywords

Data, Information, Knowledge, Ontology, Object, Object Model, ONR, SILS, UML

Introduction

The Shipboard Integration of Logistics Systems (SILS) is a concept developed by Dr. Phillip Abraham of ONR. SILS can be explained as integrating shipboard and supporting shore side systems for information sharing as an enabling platform for the development of key technologies geared towards providing much more efficient and timely logistics. This in turn provides higher levels of mission readiness that are sustainable over longer periods. Examples of these technologies include: intelligent software agents, predictive failure technologies, distance support, and self-sensing diagnostic capabilities. Many of these technologies require information sharing to derive, share, and apply the dynamically growing bodies of knowledge that they embody to the problem of naval logistics.

The SILS concept is manifest in the design of a series of decision support systems based on the Integrated Collaborative Decision Model (ICDM); a collection of guiding principles, architectural components, and tools developed by the CAD Research Center for the implementation of agent based decision support systems. These systems are as follows:

1. The Collaborative Agent-based Control and Help System (COACH) assists naval servicemen in the performance of time critical repairs.
2. The Ordnance Tracking and Information System (OTIS) assists ordnance officers in the planning, tracking, and implementation of ordnance movements aboard aircraft carriers.
3. The Mission Readiness Analysis Toolkit (MRAT) assists the commanding officer of Navy ships and their department heads in assessing and preparing the readiness of their ship for combat.

This paper begins by providing some background information covering the concept of an ontology and the progressive evolution of a single ontological model framework developed in the context of the three SILS decision support systems: COACH, OTIS, and MRAT. The conventions employed in the text and figures of the paper are also provided. This is followed by a description of the overarching framework that distinguishes and relates the concepts of data, information, and knowledge in the context of SILS. Next, successive sections provide general descriptions of data, information, and knowledge, and introduce the corresponding top-level model elements, derived directly from the abstract conceptualization that the framework provides for each. Then an example is provided that describes the application of the generalized SILS model to represent and reason about a specific real world problem. The paper concludes by making observations about the nature of data, information, and knowledge by utilizing the presented model and example as an environment for their contemplation.

Background

An ontology is a conceptual model of the world that can be used to create virtual emulations of it within the bounds of the application domains of a particular system. Concepts in common between these three ICDM systems are all implemented with the same ontological elements. To this common core, non-overlapping extensions that address the concepts unique to an individual system are added. Information sharing amongst systems is accomplished most efficiently and accurately with common ontological elements. By sharing common ontological elements, translation is not required for these systems to exchange information, which results in a more efficient exchange without ambiguity or lose of information.

COACH presented the ontology development team with a domain that potentially included all the equipment in the U.S. Navy. The team looked to apply the concepts espoused by Martin Fowler in his book Analysis Patterns, reusable object models (Fowler 1997a) to develop models independent of any particular piece of equipment that could be extended at runtime by users in the field to deal with new types of military equipment or variations on existing types as they were introduced. A key feature of the COACH model was the use of a knowledge instance model to allow the ontology to be extended at runtime. With OTIS the team found a large percentage of the precompiled class model from COACH could be reused, given a knowledge instance model to tailor it to the domain of ordnance handling aboard U.S. Navy aircraft carriers. During the implementation of OTIS the model, the concepts developed for the COACH model evolved resulting in a formalized split between the operational level of the model and the knowledge level. With SILS, the team was again able to reuse large portions of the evolving generic ontology. SILS introduced a new aspect in that it is primarily driven by external system

data inputs. This resulted in the formalized incorporation of the concept of data in the model framework. The concepts associated with the OTIS operational level also evolved to result in a model framework that formally distinguishes the individual concepts of data, information, and knowledge.

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 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. Pertinent characteristics of the different constructs are described as the constructs are introduced throughout the document. 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 and quotes object instance names.

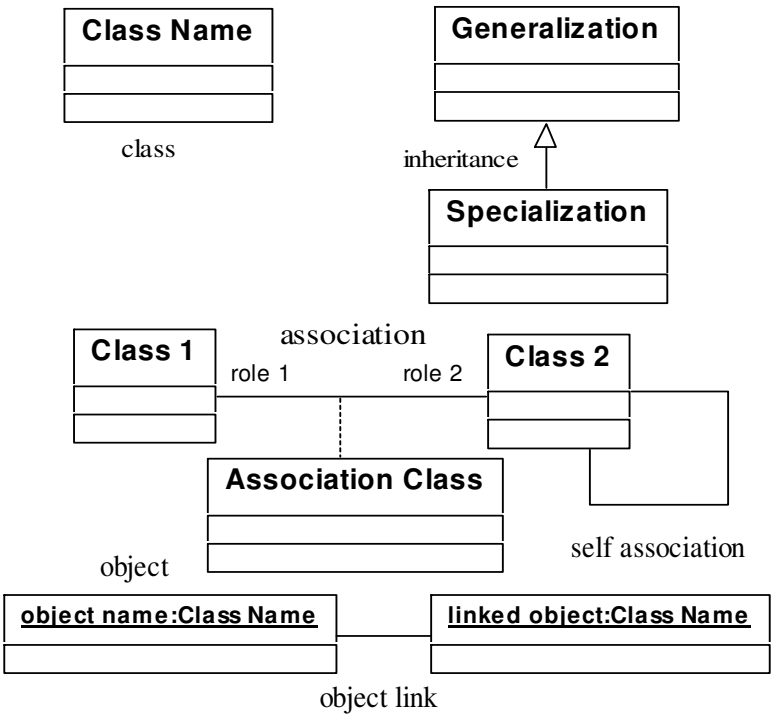


Figure 1 UML Notions Employed in this Paper

SILS Model Framework

The SILS Model Framework, depicted in Figure 2, provides top-level concepts and structure for the development and implementation of ontologies intended to model the core problem domain for the encompassing decision support system. It is rooted in a higher-level system framework

that provides structural representation for those elements of decision support system implementation that are independent of any particular domain. Examples include concepts such as *User*, *Access Permission*, and *Session*. *Sessions* are used to partition *Domain Objects* into disjoint worlds (sets) to support such things as training scenarios and what if experiments independent from the primary operational picture, but within the same conceptual and physical system environment. All objects used to represent the core problem domain of the system are *Domain Objects*, which serves as the common base class that roots the SILS Model Framework and captures all relationships with the external context provided by the SILS System Framework.

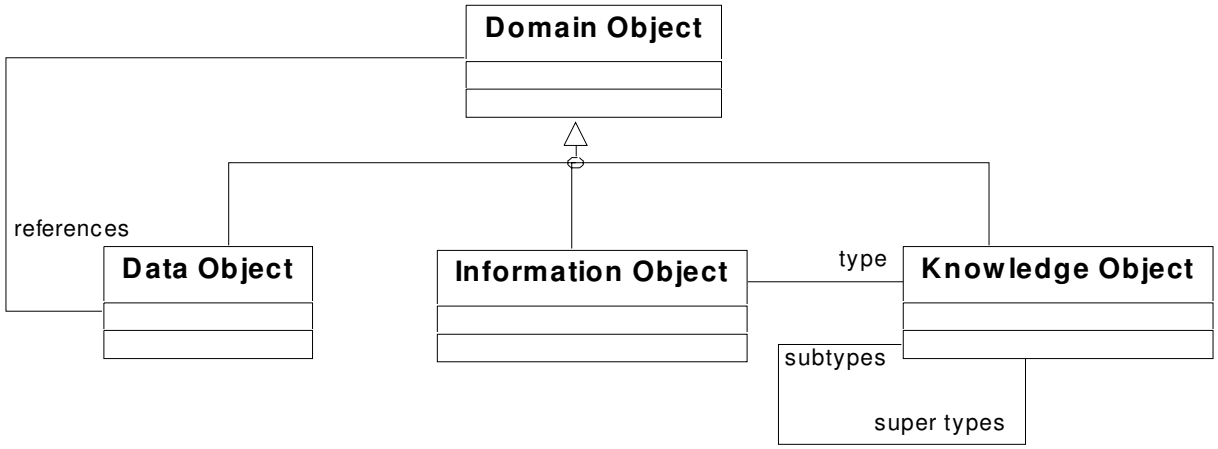


Figure 2 Model Framework

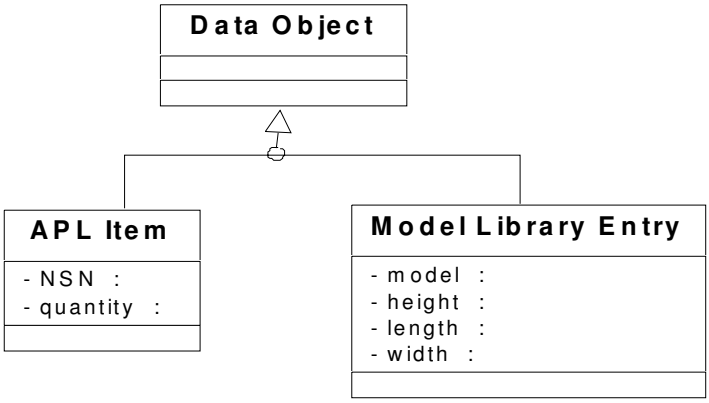
In addition to external context, *Domain Object* supports those characteristics of the representation (ontology) and implementation (object model) shared by all objects within the problem domain representation. The model defines three types of *Domain Object*: *Data Object, Information Object, and Knowledge Object*, which respectively model the distinct real world concepts of data, information, and knowledge within the virtual context of a SILS system implementation. The model defines three associations to capture the key relationships between these concepts.

All *Information Objects* have an association with at least one *Knowledge Object* that defines the logical type or types of the *Information Object.* This will be referred to as the knowledge level type of the *Information Object* to distinguish it from the standard classification mechanism indicated by the closed arrow in a UML diagram. *Information Objects* use their knowledge level type in conjunction with the standard classification mechanism to augment the fixed meta-model provided by the object oriented implementation environment. Using this approach the knowledge level serves as a dynamic meta-model that supports meta-level relationships, and allows for dynamic and multiple classification schemes that support runtime extensions. The subtype supertype association is used to compose *Knowledge Objects* into classification taxonomies upon which much reasoning can take place. Data objects are used to hold standardized reference data defined by the DOD or imported from external systems. *Domain Objects* may associate *Data Objects* in order to supplement the information carried by their attributes. In this manner, the associated data may be elevated to the level of information.

SILS Data Model

Data consists of words or numbers without relationships; thereby, requiring the context to be inferred to provide meaning. A random stream of words or numbers can be thought of as raw data while a fixed sequence of words and numbers can be thought of as structured data. Structured data is the primary building block upon which information is built; however, the data must be structured to correspond to the real world entities within the domain in order to build information. Given structured data corresponding to entities within the domain, the transition to information is made by adding the relationships between entities to pin down the situational context to such an extent that no inferences are required to provide meaning. These ideas are discussed in depth in (Pohl 2000). Data is the primary means by which information and knowledge are communicated as the software systems, agents, or human beings participating in the communication typically have differing ontologies. Differing ontologies necessitates the use of inferences by the sender to decide what to send and by the receiver to translate it into his internal ontology from which he makes sense of the world in which he lives.

Figure 3 Data Model Fragment



The SILS Data Model Fragment shows two specializations of *Data Object*: *APL Item* and *Model Library Entry*. *Data Object* is an element of the SILS Model Framework that is depicted in Figure 2. APL Item represents an entry in the Allowance Parts List for a Navy Ship. This list specifies the type, by National Stock Number (NSN) and quantity of the spare parts to be carried aboard ship. When represented as data a human or software agent must already know what ship the APL corresponds too, perhaps by assuming it references the ship upon which the data server is resident. By linking a specific APL Item to a specific ship, perhaps by using the *Domain Object* reference association, an inference need not be made. *Model Library Entry* is a line item in a standardized US Army Catalog of vehicle and equipment dimensions. An experienced human operator knows that models map to National Stock Numbers (NSNs) and could therefore use a *Model to NSN Map Entry Data Object* (not shown) to correlate *APL Item* objects to *Model Library Entry* objects in order to access the dimensions of an *APL Item*. By associating APL Item objects to Model Library Entry objects the dimensions of an APL Item could be directly obtained without the inferences and data joins previously required; thereby, providing information rather than data in this instance.

SILS Information Model

This section describes the simplified top-level fragment of the SILS Object Model for representing information that is depicted in Figure 4. Information consists of structured data with relationships defined by an explicit conceptualization known as an ontology. The relationships provide context and thus meaning. Information is declarative in nature and thus associated with the present and the past.

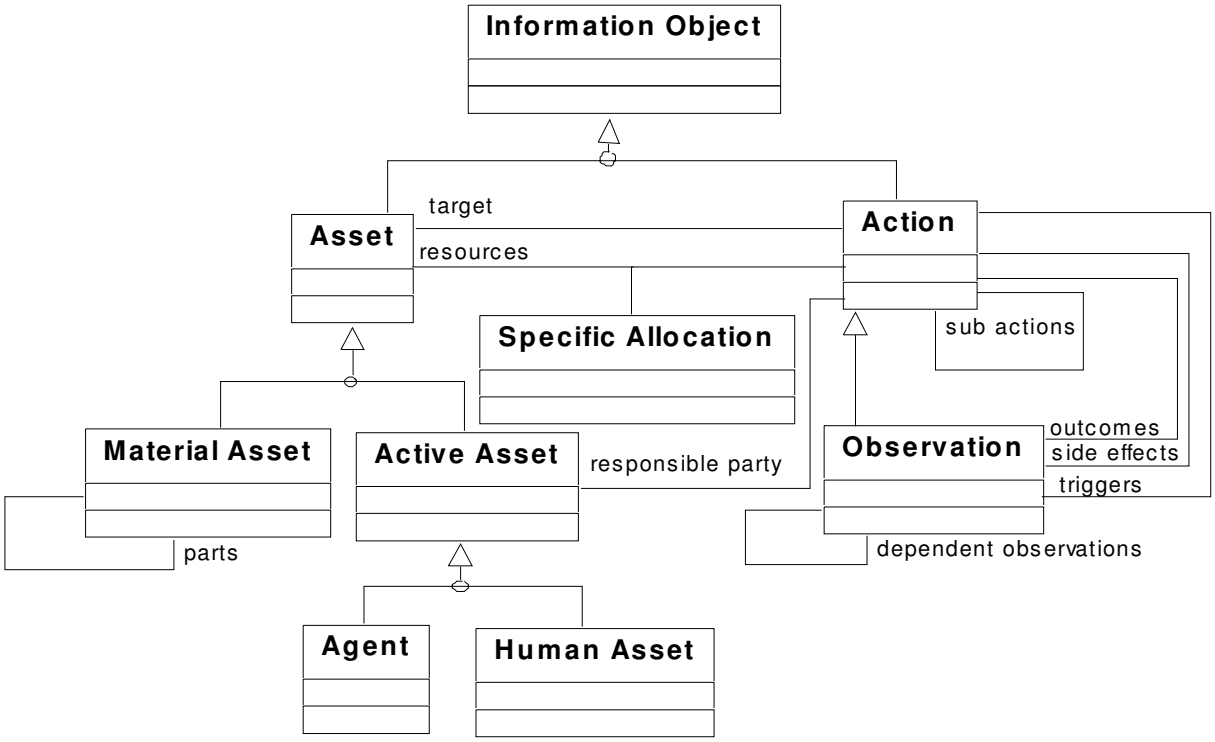


Figure 4 Information Model Fragment

This model fragment expands the concept of information in the context of SILS by defining concrete specializations of *Information Object*. *Information Object* is an element of the SILS Model Framework that is depicted in Figure 2. Key to understanding the SILS Information Model is the concept that each *Information Object* has a knowledge level type that provides it with logical meaning as discussed in SILS Model Framework section.

The SILS Information Model Fragment defines three primary types of *Information Object*: *Asset*, *Action*, and *Observation*. An *Action* is used to represent the performance of a *Protocol* as indicated by the associated knowledge level type to a *Protocol Knowledge Object.* Key attributes of *Action* (not shown) are the start time and end time, which may be actual or planned depending on the value of the status attribute of *Action.* An *Action* may be further decomposed in to sub actions as indicated by the association with itself.

An *Asset* is used to represent concrete entities within the domain the logical type of which is indicated by the knowledge level type association to an *Asset Type*. Two types of *Asset* are defined: *Material Asset* and *Active Asset*. *Material Assets* associate with *Asset Types* that define

things such as ships, guns, uniforms, petroleum products, and types of food. A *Material Asset* may contain parts, which are themselves *Material Assets* as indicated by the association with itself. *Active Assets* are things that may act to change the environment. Two types of *Active Asset* are defined: *Human Asset*, which may be a *Person* or an *Organization*, and *Agent*, which corresponds to a software-based entity.

An *Observation* is used to represent the occurrence of some *Phenomenon* within the domain as indicated by the associated knowledge level type to a *Phenomenon Knowledge Object*. Note that an *Observation* is an *Action* and therefore has all the characteristics and relationships indicated for an *Action*. An *Observation* may indicate the *Observations* upon which it depends by the association with itself. This is useful for inferred *Observations* as it allows the *Observations* upon which an inference was based to be recorded. If one or more of the dependent *Observations* are invalidated, it may be an indication that the inferred *Observation* needs to be invalidated as well.

Now that the individual object types defined in the SILS Information Model Fragment have been described, meaningful descriptions of the associated relationships can be provided. Looking again at *Action* it can be seen that an *Action* may optionally (note that association multiplicities are not depicted here for the sake of compactness) be performed on an *Asset*, and may indicate resources used or allocated, depending on the value of the Action type attribute, during or for the execution of the *Action*. Resources are associated by means of an association class, which allows attributes to be tied to the association itself. This allows, for example, the representation of the allocation of an *Asset* to an *Action* for a block of time that differs from the block of time over which the *Action* occurs. Associations also indicate the *Active Object*: *Person*, *Organization*, or *Agent* responsible for performing it. With the addition of these relationships, one can see how the individual sets of structured data represented by *Action* and *Asset* are elevated to the level of information.

The transformation to information becomes even more apparent when the associations between *Action* and *Observation* are examined. *Action* triggers allow the *Observations* that prompted the *Action* to be recorded. For example, an *Observation* of the phenomenon 'broken' on generator number two can be recorded as the trigger for an *Action* using the *Protocol* 'diagnose generator'. Side effects and results can also be indicated. Continuing on the previous example it can be indicated that the *Action* to diagnose the generator resulted in ship power circuit number 4 being shut down from 4 to 5 o'clock as indicated by an *Observation* with start time 4 and end time 5 on the *Asset* 'power circuit number 4' of the *Phenomenon* 'shut down'. Finally, a result could be indicated by a result association (link) to an *Observation* of the 'generator main bearing' of the *Phenomenon* 'requires replacement'.

SILS Knowledge Model

This section describes the simplified top-level fragment of the SILS Object Model for representing knowledge that is depicted in Figure 5. *Knowledge Object* is an element of the SILS Model Framework that is depicted in Figure 2. Knowledge consists of information-based inferences that are predictive in nature and thus associated with the future.

This model fragment expands the concept of knowledge in the context of SILS by defining three concrete specializations of *Knowledge Object*: *Asset Type*, *Protocol*, and *Phenomenon* that serve as the respective targets of knowledge level type associations for the *Information Objects*: *Asset*, *Action*, and *Observation*.

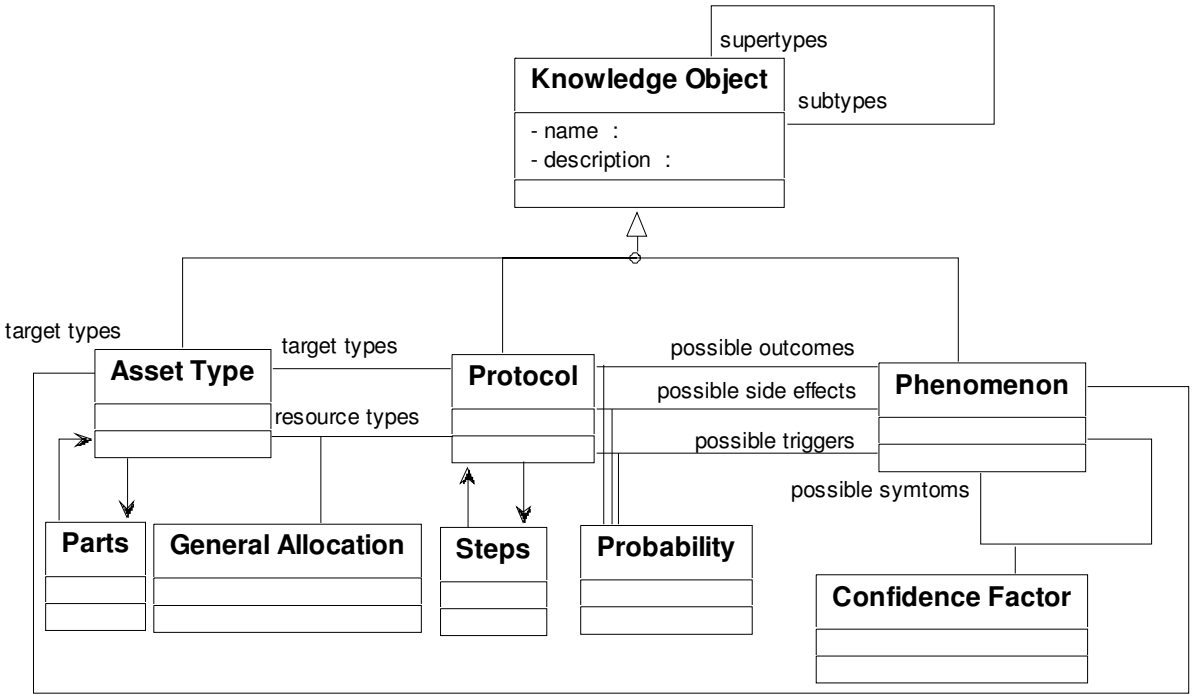


Figure 5 Knowledge Model Fragment

A *Protocol* is used to represent standard procedures or processes independent of the *Actions* that employ them. It may be further decomposed into steps as indicated by the association with itself through the steps collection class. *Protocol* steps, which are also *Protocol* objects, mirror the sub actions of *Action*, but cannot be implemented with self-association, which requires referenced objects to be unique. Every *Action*, including sub actions, is unique while the same *Protocol* could be used repeatedly to build up a higher-level *Protocol*. For instance, the *Protocol* 'rotate tires' may use the unique *Protocol* 'remove wheel' four times. If the *Action* object that represents the execution of the 'rotate tires' *Protocol* is decomposed in the same manner (typical but not required), it would associate as sub actions four unique *Actions*, performed at different times or by different people, that all reference the same 'remove wheel' *Protocol.* *Protocol* uses the target types association to indicate the applicable *Asset Types*. The *Asset Types* required to implement it are indicated by the *General Allocation* association class, which provides attributes to indicate quantity and time.

An *Asset Type* is used to represent the types of concrete entities within the domain. Just as a specific *Material Asset* object may contain specific parts, a particular *Asset Type* may contain a collection of part types. Similar to the situation with *Protocol*, *Asset Type* cannot use self-association to indicate part types due to the uniqueness constraint on associations; however, since

part order does not matter only one entry per type of part along with an indication of quantity is required.

Another point of interest involves the absence of classes derived from *Asset Type* in the knowledge model in contrast to the derivation tree off *Asset* in the information model which contains four classes: *Material Asset*, *Active Asset*, *Human Asset*, and *Agent*. The implementation paradigm employed by the SILS Model does not utilize the inheritance mechanisms provided by object oriented implementation languages as the primary classification mechanism for *Information Objects*. The language provided mechanism is used only to add the attributes or associations required by the implementation. Logical classification for *Information Objects* is provided by constrained associations to object instances in the knowledge level. While logical classification for *Knowledge Objects* is implemented by linking object instances using the subtypes supertypes association of *Knowledge Object*.

A Phenomenon is used to represent things that can be observed within the domain. Just as an *Observation* may indicate dependent *Observations* though self-association, *Phenomenon* objects may indicate symptomatic information. For example, radio (*Asset Type*) 'does not work' (*Phenomenon*) could be symptomatically linked to battery (*Asset Type*) 'is dead' (*Phenomenon*). As with *Asset Type* and *Protocol*, a more complex self-association mechanism is employed for *Phenomenon* in the knowledge model than for *Observation* in the information model, but for a different reason. In the information model the *Observations* used to infer another *Observation* may be noted. In the knowledge model, it is the probability that one *Phenomenon* infers another that is being noted. In this case, uniqueness is not an issue so an association class is employed to add a probabilistic attribute such as a confidence factor to the symptomatic links between *Phenomenon* objects. This type of probabilistic association is common to the knowledge model.

Protocol and *Phenomenon* have similar relationships to those between *Action* and *Observation* in the information model. Where associations between *Action* and *Observation* provide triggers, side effects, and outcomes, associations between *Protocol* and *Phenomenon* provide possible triggers, possible side effects, and possible outcomes. All of these knowledge model associations utilize an association class to provide probabilistic measures of the individual possibilities. This paper presents a simplified model for the sake of brevity and understanding. The actual SILS model provides additional representational machinery for grouping outcomes side effects, and triggers in order to indicate things like the outcome will be X half of the time or Y and Z the other half of the time.

Object Instance Example

This section presents a simple example intended to clarify the concepts presented thus far. The first step in tailoring the SILS object model to a particular domain is to develop a knowledge instance model from the knowledge model classes. Rather than providing a knowledge instance model fragment from one of the SILS systems: COACH, OTIS, or MRAT, this example will apply the SILS model to the domain of medical diagnosis, which a broader audience can easily relate too.

First, a phenomenological taxonomy is developed using the subtypes super types association of inherent to all *Knowledge Objects.* Note that the classes in the SILS object models are not reasoned on directly, but rather serve as templates for creating instances upon which reasoning may occur. A simple taxonomy and the knowledge model classes from which it was derived are shown in Figure 6. This hierarchy shows that an 'Infection' is a type of 'illness' and that an 'Infection' may be either a 'Bacterial Infection' or a 'Viral Infection'. In order to demonstrate the ability for reasoning on the taxonomic structures common to the representation of knowledge, consider a presence *Observation* (an *Observation* can record the presence or absence of a *Phenomenon*) posted for a person named John (information model *Human Asset* instance) on the 'Infection' *Phenomenon.* Also, consider an absence Observation posted for a person named Jane on the same *Phenomenon*. Presence observations propagate up the tree so that an intelligent agent can easily conclude that not only does John have an infection he has an illness as well and all that it entails; however it is not know it the infection is bacterial or viral. Similarly, absence observations propagate down the tree. In the context of the example, it is observed that Jane does not have an 'Infection' so one can say Jane does not have a 'Bacterial Infection' or 'Viral Infection' as well; however, at this point is cannot be determined whether or not she has an 'illness'.

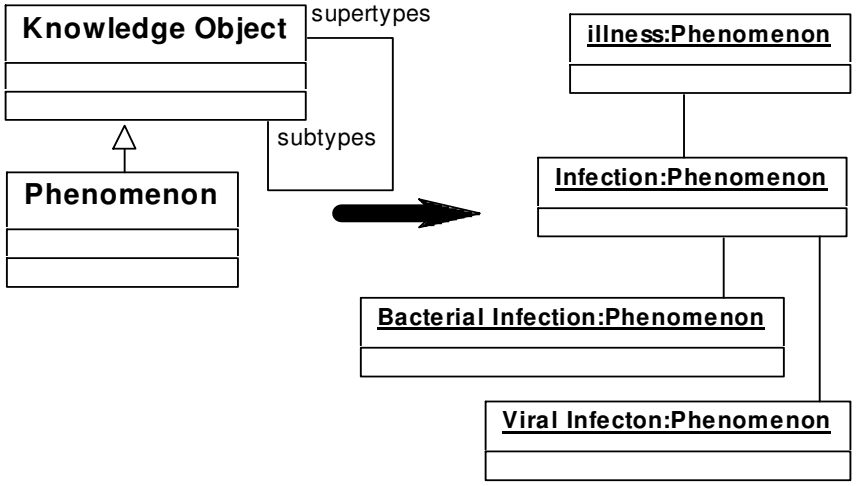
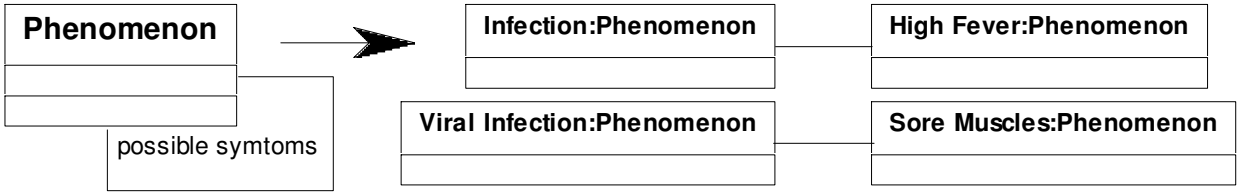


Figure 6 Phenomenological Hierarchy

The next step is to tie these phenomena to symptomatic information. Note that a symptom is also a *Phenomenon* that may be resident is some taxonomy. In this example, 'Infection' is linked to 'High Fever' and 'Viral Infection' is linked to 'Sore Muscles' as show in Figure 7. The probabilistic information associated with the links and the role names has been left off for simplicity's sake. Note that since a 'Viral infection' is also an 'Infection' as indicated by the super types subtypes links depicted in Figure 6. This relationship indicates that that a 'Viral Infection' shares the characteristics of an 'Infection' or in this particular case also has 'High



Fever' as a symptom, but adds new characteristics, the symptom of 'Sore Muscles' in this case, which are not characteristics of the *Phenomenon* 'Infection'.

Figure 7 Symptomatic Links

To finish the knowledge instance model portion of the example that tailors the SILS Object Model to a limited portion of the domain of medical diagnosis, the representative *Protocol* objects: 'Measure Body Temperature' and 'Check for Sore Muscles' are defined. The *Phenomenon* 'Illness' is linked as a trigger to the Protocol 'Measure Body Temperature', and the *Phenomenon* 'Infection' is linked as a trigger to the *Protocol* 'Check for Sore Muscles' as shown in Figure 8. As before, the super type subtype link between *Phenomenon* objects 'illness' and 'Infection' indicate that the *Protocol* 'Check for High Fever' is triggered by the *Phenomenon* 'Infection' as well as 'illness'; however, only the *Phenomenon* 'Infection' is a trigger for the *Protocol* 'Check for Sore Muscles'.

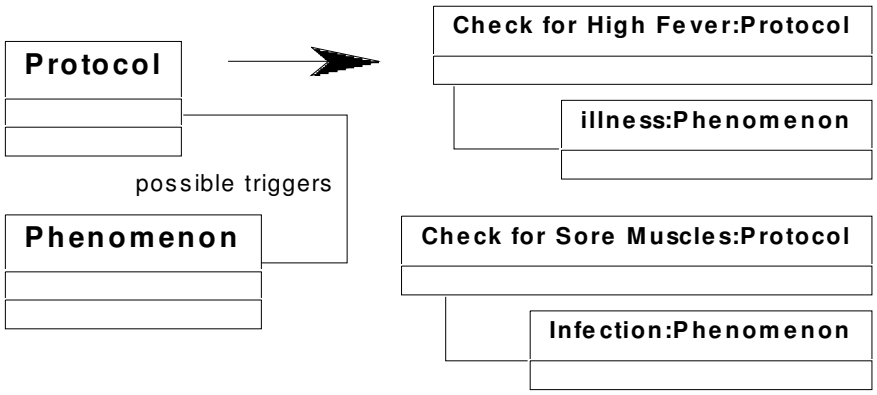


Figure 8 Protocol Triggers

With a knowledge instance model defined, the SILS Information Model can be used to record information and reason on this limited domain of medical diagnosis. First, the example records an observation that the *Person*, where *Person* is a SILS Information Model specialization of *Human Asset*, Mike Zang has indicated that he is feeling ill as shown in Figure 9.

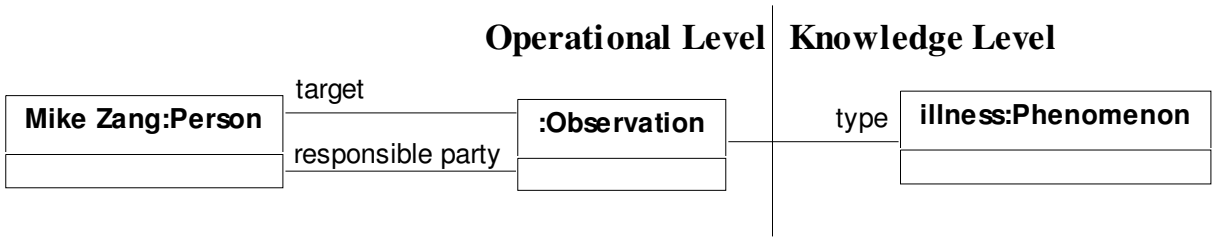


Figure 9 Illness Observation

A person named John Smith then measures the body temperature of Mike, prompted by the triggering *Phenomenon* of 'illness' for the *Protocol* 'Measure Body Temperature' in the knowledge instance model of the example, as shown in Figure 8. The outcome of this *Action* is the *Observation* by John Smith that Mike Zang has a 'High Fever'. These information postings are shown in Figure 10.

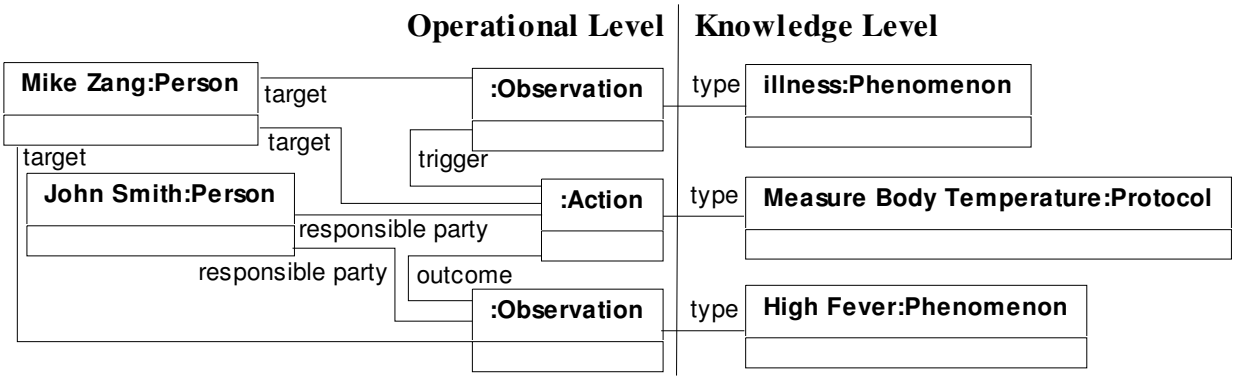


Figure 10 High Fever Observation

'High Fever' is indicated as a symptom of 'Infection' (see Figure 7), a trigger (see Figure 8) that prompts John Smith to make an *Observation* of 'Infection' then perform the *Action* of 'Check for Sore Muscles' on Mike with the outcome *Observation* that Mike does indeed have 'Sore Muscles'. These information postings are shown in Figure 11.

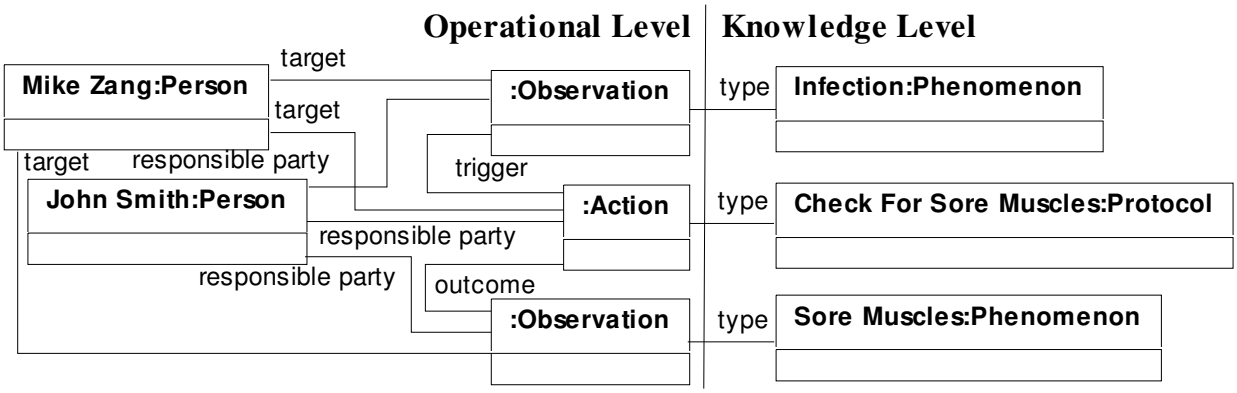


Figure 11 Sore Muscles Observation

Using the symptomatic knowledge that 'High Fever' and 'Sore Muscles' indicate a 'Viral Infection' (shown in Figure 7), a 'Medical Diagnostic' *Agent* posts an *Observation* that it thinks Mike Zang has a 'Viral Infection', as shown in Figure 12.

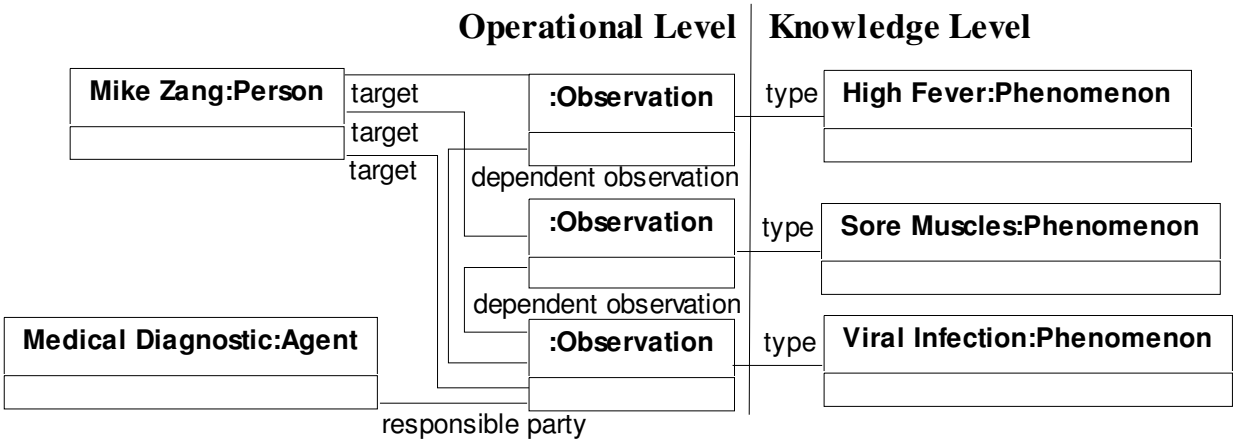


Figure 12 Agent Diagnosis of Viral Infection

Observations

While some standard characteristics of the individual concepts of data, information, and knowledge have been provided, it can still be difficult for one to precisely pin down the differences. This is partially due to the fact that their nature can vary due to the context in which the terms are applied. The clean separation of data, information, and knowledge within the SILS Object Model provides an excellent environment to glean insightful observations about the nature of these concepts in the context of SILS that may carry over to other environments as well. This section concludes the paper by listing and describing some of the observations made by the author as his team developed the SILS model and he worked to describe it in the previous sections of this paper. The reader is encouraged to play with the model by extending it or making up example instances to see what additional observations can be derived.

Knowledge is derived from information. As *Actions* are resourced through associations to *Assets*, the corresponding *Asset Types* can be associated to the *Protocol* of the *Action* or the probabilities of the existing link can be modified using Bayes Law for example. In this manner, knowledge can be learned by observing operational information.

Information is dependent on knowledge for meaning. An *Action* or *Observation* is meaningless without the associated *Protocol* or *Phenomenon*. One could imply that something was done in the case of an *Action* with no knowledge level type but what was done could not be implied. Similarly one could imply that something was seen, measured, or inferred in the case of an *Observation* with no knowledge level type but what that something was could not be implied.

Information is a simplified reflection of knowledge. One can see the close parallels between the information and knowledge model fragments. While it appears the information model fragment has more classes, one must remember that the knowledge level of the model acts as a meta-level for information. While there is only one *Action* class on the operational side of the model there are a practically limitless number of *Protocols* with which *Action* objects may associate.

Knowledge exhibits more complex associations than information. This observation is exhibited in two ways. First, self associations in the knowledge model must typically be implemented with reference objects or association lists as they may often associate the same object more than once, while the self associations in the information model can use the standard set implementation, as it is rarely the case that an information object needs to associate with itself. Second, associations between knowledge objects are typically probabilistic while those between information objects are direct.

Knowledge is more dependent on the domain than information. *Actions*, *Observations*, and *Assets* apply to just about any domain one can think of while the corresponding knowledge level entities *Protocol*, *Phenomenon*, and *Asset Type* can vary drastically from one domain to another.
Information recorded in error should be invalidated rather than destroyed or modified. This becomes evident when using the model. Consider the medical diagnosis example. Once the 'viral infection' diagnosis has been made it is likely this will be linked as a trigger for some sort of treatment *Action*. If the diagnosis turns out to be wrong it cannot simply be modified to

point to a new *Protocol* or even destroyed, as the reasoning behind the treatment must be retained to justify the treatment that could now be incorrect given the new information.

Invalid knowledge should be temporally modified or destroyed. This only applies to temporal systems that allow one to go back in time. Take the diagnostic example again. Diagnoses are made based on the knowledge of the times, which can change dramatically over time particularly in the area of medical diagnostics. This can be dealt with if all knowledge level objects and association classes have activation and deactivation dates, for example. In this manner, one can correctly judge past decisions based on the knowledge available at the time the decisions were made.

New information refines knowledge. This is evident in the probabilistic associations of the knowledge level. If a certain *Protocol* provides the option of two different types of tool with which to implement it, every time an *Action* records the use of tool X instead of tool Y, the probability that associates the type of tool X to the *Protocol* increases while that associated with the type of tool Y decreases.

New knowledge allows a more precise specification of information. Consider again the medical diagnosis example. The Protocol 'Measure Body Temperature' could be extended with new knowledge that partitions 'Measure Body Temperature' into 'Measure Body Temperature Orally', and 'Measure Body Temperature Aurally' each of which has different margins for error. Using one of these new knowledge level *Protocols*, the example action on 'Measure Body Temperature' could be more precisely specified.

New information is easily identified whereas new knowledge is not. New information is always unique by definition. But consider for example a user of the model wishing to post an observation that his car engine is 'busted'; he does not find an existing 'busted' *Phenomenon* and therefore adds a new one. However the Phenomenon 'broken' did exist. Are 'broken' and 'busted' the same? It is hard to say although a detailed study of the associated knowledge could help. For example, do they apply to the same set of *Asset Types*? This is a difficult issue to solve particularly if the associated system requires support for knowledge level extensions by the end users.

Information does not combine like knowledge. Information by definition is unique so that if two equally large collections of information are combined the result is simply twice the amount of information. Combining knowledge is much more complicated as it first involves identifying identical pieces of knowledge, particularly if dynamic extensions have been allowed, and then involves the combining of the probabilistic data associated with knowledge level associations.

Data referenced by information becomes information. Since an *Information Object* is placed in context by the associations it exhibits, data references by it will be placed in the same or similar context and are therefore raised to the level of information.

Standardized Data is more appropriately linked to Knowledge than to Information. The standardized reference data employed by the DOD and other organizations capture general not

specific characteristics. Consider the *Model Library Entry* example, an example entry might provide dimensions for a model of vehicle, say a humvee in the ambulance configuration. A humvee is not a specific *Asset* (operational level) as in Mike's humvee but an *Asset Type* (knowledge level).

References

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

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

Pohl, Jens (2000); Transition from Data to Information; Collaborative Agent Design Research Center, Cal Poly San Luis Obispo, Ca