# Functional Ontology Artifacts: Existent and Emergent Knowledge

Daniel E. O'Leary*
*University of Southern California, 3660 Trousdale Parkway, Los Angeles, California 90089-1421*

Ontologies have primarily been promoted to facilitate interagent communication and knowledge reuse. There has not been as much emphasis on using ontologies to improve system quality. As a result, typically, ontology development, and verification and validation are treated as different stages in the life cycle. However, this paper argues that ontology design should include emergent knowledge that previously might only have been considered or generated at the time of verification and validation. Emergent knowledge differs from the existent knowledge that is typically included in ontologies. Existent variable knowledge is knowledge about variables that derives from the model of the variable being used, e.g., how is a conceptual variable measured. Emergent variable knowledge is knowledge about variables that emerges after variables have been named, e.g., cardinality, which variables interact with each other and how, e.g., independent and dependent variables. Much emergent knowledge can be used for verification and validation. Including emergent knowledge can facilitate the use of ontologies to design and to build systems with fewer anomalies prior to testing. 

## I. INTRODUCTION

Ontologies are a relatively new and emerging area of research. Where ontologies are often seen as one of the "first steps" in system development and verification and validation are typically seen as one of the "last steps." Ontology is typically seen as a design issue, while verification and validation are seen as testing issues. As a result, ontologies have received little attention in the area of verification and validation, and conversely. In particular, verification and validation researchers primarily focused on treating ontologies as an artifact (e.g., Ref. 1) that can be verified and validated. Thus, much of the previous literature has been focused on designing tests for ontology artifacts, e.g., frame or object representations.

Ontology specification is not yet a science. There are few bounds as to what should or should not be included in an ontology. In part, this is because of the importance of the domain in ontology development and because of the recent-

* e-mail: oleary@usc.edu.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 16, 411–424 (2001)

ness of the development of ontologies. Further, most emphasis on ontologies has focused on multiple agents and knowledge reuse, not system quality. As a result, this paper provides one approach to developing an ontology and elicits some knowledge for inclusion in ontologies that have not received much attention.

Existent variable knowledge is knowledge about variables that derives from the model of the variable being used. For example, how a variable is measured is existent knowledge. Emergent variable knowledge is knowledge about variables that emerges after variables have been named. For example, the number of times a variable name appears in a model or the tuples of variables that are paired to generate knowledge statements are emergent knowledge.

The source for the additional ontology information derives from distinguishing between existent and emergent knowledge. Typically, ontology definition includes existent knowledge only. Including emergent knowledge is important, since as with all computational artifacts, the more about an ontology that can be structured, specified, or constrained, the more the ontology can be tested to determine if it meets the needs of that structure or those specifications and constraints. Further, the more structure that can be specified in the ontology, the more that the system can be investigated to ensure that it meets those specifications and constraints. As a result, this article is designed to elicit additional knowledge that can be captured and employed in the ontology.

In particular, this article argues that ontologies also should include emergent characteristics, many of which provide a basis for ensuring artifact quality and for artifact testing. In so doing, this article addresses the following questions:

- What can we build into an ontology that facilitates system quality and correctness, not just reuse and communication?
- What properties of ontology artifacts facilitate verification and validation of that resulting ontology?

This article extends previous research on ontologies, and verification and validation with two primary contributions. First, this article extends the notion of ontology to include emergent knowledge. Second, this article pushes activity that previously would have occurred with system verification into the specification of the ontology and system.

To elicit those contributions, this article proceeds as follows. Section II provides some background on ontologies, briefly reviewing what information is in an ontology, when ontologies are required, and how they are represented. Section III provides a review of the previous verification and validation ontology literature and an overview of the example used as a case study to illustrate the analysis and types of knowledge that can be captured in the ontology. Section IV investigates ontological specification of variable components, while Section V analyzes ontological specification of variables. Section VI discusses specification of variable roles that can be used to facilitate ontology structuring. Section VII develops ontological specification of variable relationships. Section VIII analyzes the use of cardinality as part of an ontology. Section IX investigates the

role of "emergent knowledge" in ontologies. Section X briefly summarizes the article.

## II. BACKGROUND: ONTOLOGIES

An *ontology* is an explicit specification of a conceptualization.[2] It is a knowledge-based specification that typically describes a taxonomy of the tasks that define the knowledge. Ontologies typically are explicit specifications of discourse between multiple agents in the form of a shared vocabulary. However, ontologies can be used in single agent systems as a means of specifying system characteristics.

### A. Ontologies are Necessary in Multiple Agent Systems

Ontologies are critical for the development of multiple agent systems so that the agents can engage in meaningful dialogues. One example of that approach was given with a discussion of the development of the Palo Alto Collaborative Testbed (PACT) system by Cutkosky et al.[3] who noted

> ...the agents involved in any transaction must agree on a common ontology, which defines a standard vocabulary for describing time-varying behavior under each view of time that is needed. What went on behind the scenes in PACT, and is not represented in computational form at all, was a careful negotiation among system developers to devise the specific pairwise ontology that enabled their systems to cooperate. The developers met and emulated how their respective systems might discuss, say, the ramifications of increasing motor size. In this fashion, they ascertained and agreed upon what information had to be exchanged, and how it would be represented.

### B. Knowledge Reuse

Ontologies provide a basis for re-using knowledge, which has a number of advantages. Previously used knowledge may not require verification and validation. Previously used knowledge does not have to be rediscovered, saving both time and effort. Previously used knowledge can be shared with others, in multiple agent systems.

### C. *A Priori* Formal Specification of Ontologies?

It is tempting to suggest that ontology development is a "formal specification" task, largely occurring before system development. However, in real world settings, ontologies often seem to emerge as the system is developed, oftentimes using relatively informal processes. For example, Cutkosky et al.[3] found that

> ...what PACT actually demonstrates is a mechanism for distributing reasoning, not a mechanism for automatically building and sharing a design model. The model sharing in PACT, as in other efforts, is still implicit—not given in a formal specification enforced in software. The ontology for PACT was documented informally in email messages among developers of the interacting tools.

That is not to say that there is no benefit to formally specifying the ontology. However, it does suggest that ontology knowledge, in part, emerges as the system is developed and as parts of the ontology become better understood.

With formal specification of the ontology either as an *a priori* specification or as a part of the system development, comes an increased ability to use the ontology for verification purposes: the more constraints on the system the more ways to tell if it is correct.

### D. What Information or Knowledge Is Contained in an Ontology?

What information is contained in an ontology? Typically, an ontology contains measurement specification and representation standards. Further, ontologies contain detailed variable expressions and relationships between variables. In particular, as noted by Cutkosky et al.[3] there is a wide range of information that depends heavily on the nature of the task being modeled.

> Agreements must be reached about concepts in the natural world, such as position, time, shape, behavior, sensors, and motors. For each concept, agreement is required on many levels, ranging from what it means to how it is represented. For instance, how should two agents exchange information about the voltage on a wire (what units, what granularity of time?); how should manipulator dynamics be modeled (as simultaneous equations or functions? in what coordinate frame?). The four systems comprising PACT used various coordinate systems and several distinct representations of time (e.g., discrete events, points in continuous time, intervals of continuous time, piecewise approximations). These representations were chosen for valid task and context-dependent reasons. They cannot simply be replaced by one standard product model (e.g., representation of time).

### E. Representation of Ontology Knowledge

Ultimately, ontology specification of variables and their relationships can be specified in a wide range of traditional knowledge representations forms, including rules, frames (e.g., Ref. 4), and networks (e.g., Ref. 5). In addition, database representations including, simple vector–table databases, trees, relational databases, or other approaches can be used. For example, in the expert system shell M.4, variable values can be listed as part of a "legalvals" statement. In the following example, the variable "industry" can take two values "insurance" and "chemical." Then

$$\text{legalvals}(\text{industry}) = [\text{insurance}, \text{chemical}]$$

Alternatively, knowledge could be represented as a tree or equivalent database representation, as in Figure 1 that describes characteristics of movement, discussed later in the paper.

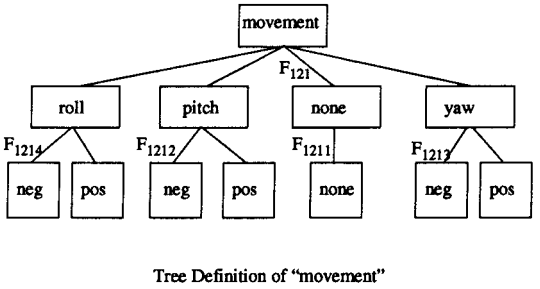


**Figure 1.** Tree definition of movement.

## III. PREVIOUS RESEARCH ON VERIFICATION AND ONTOLOGIES

There has been limited research joining notions of ontologies and verification and validation research. In part, the research to date, is probably limited for two primary reasons. First, the formal notion of ontology in knowledge-based systems is relatively recent (e.g., Refs. 2 and 3). Second, ontologies might just be viewed as another artifact or form of knowledge representation, such as rules. As another form of knowledge representation, verification and validation would exploit the structure of ontologies to satisfy classic concerns of consistency, completeness, and correctness (e.g., Ref. 6). In addition, since ontologies can be structured as traditional knowledge representations, e.g., rules, frames, semantic nets, or objects, much verification and validation of ontologies would be similar to the verification and validation of those forms of knowledge representation previously explored in the verification and validation literature at the systems level.

### A. Previous Research

Gomez-Perez[5] was one of the first to address the issues of ontologies and verification, jointly. Gomez-Perez[5] provides an investigation of evaluating an ontology in which she addresses the following questions:

- What does evaluation mean? (judge technically features).
- What does assessment mean? (judge usability and utility of features).
- What can be evaluated? (definitions, documentation, and software).
- Why evaluate? (guarantee correctness and completeness of definitions, documentation, and software).
- What to evaluate against? (competency questions, requirements, and real world).
- When to evaluate? (iteratively . . . as soon as possible).
- How to evaluate? (avoid ad hoc).
- Who evaluates? (various).
- Where to evaluate? (anywhere).

Gomez-Perez[5] then illustrated the verification of an ontology with an example drawn from Gruber's[7] "biblio text" example. In analysis of that example, Gomez-Perez pays particular attention to verification of definitions.

Waterman and Preece[4] describe use of a tool "DISCOVER" that verifies knowledge represented in COVER[8] rule language (CRL) and meta-ontology for verification of expert systems (MOVES). MOVES is a hierarchical frame-based system with multiple inheritance and constraints. DISCOVER uses a translation program to convert frame-based ontologies in MOVES into rule-based CRL. Once this has been done, mapping rules define the relationship between the terminology and the knowledge base. DISCOVER uses a slightly different anomaly checker than COVER.

### B. Example Used in this Paper

Throughout this paper, an example is used to illustrate selected ontology issues. The example, drawn from Lawler and Williams,[9] has also been discussed in Landauer[10] to illustrate rule-based system principles of correctness. This paper uses some of the work generated by Landauer[10] as the basis of analysis presented here. However, the use of the example here is for illustration of ontology-based issues. As a result, parts of the example are slightly altered to accommodate particular interests.

The example concerns a CLIPS knowledge base for fault diagnosis, isolation, and recovery of the manned maneuvering unit (MMU). The MMU is designed to move a human astronaut around in space. The system has 104 "if ... then ..." rules. Although the example is drawn from a rule-based system, no rules are analyzed. Instead, the focus is on the ontological structure underlying the semantic expression of the model.

## IV. ONTOLOGY SPECIFICATION OF VARIABLE NAME COMPONENTS

For the discussion in this paper, at the lowest level of an ontology are variable name components, not variables. Variable components will be used to create variables. When assembled, these variable components define the individual objects and their functional capabilities.

Assume that the ontology is concerned with an ordered set of objects $O = (\alpha, \beta, \chi, \delta, \ldots)$. Let $F_1(\alpha)$ indicate the representation of the object $\alpha$ at the root of the tree $\alpha$. Let $F_{1j}\ (F_1(\alpha)) = \{\alpha_{1j,1}, \ldots, \alpha_{1j,j}\}$ represent the $j$th-ordered set of the object $\alpha$ at the first level of the tree, which takes at most one of the mutually exclusive values $\alpha_{1j,1}, \ldots, \alpha_{1j,j}$. Let $F_{1jk}\ (F_{1j}\ (F_1(\alpha)))$ be an ordered set at the second level of the tree, etc. for each functionally related property, defining a tree as in Figure 2.

*Example*. The MMU knowledge has a number of objects, including, gyroscope and automatic attitude hold that are not used as variable names, but only as

components of the variable names that also include their functions. Those objects could be represented as in the original system, NAME SET 1.

*Objects*
$O$ = (gyroscope, automatic attitude hold, . . . )

$F_1$(gyroscope) = gyro
$F_1$(automatic attitude hold) = *aah*

*Functional Relationships*
$F_{11}$(gyro) = {on, off}
$F_{12}$(gyro) = {movement}
$F_{121}$(movement) = {none, pitch, roll, yaw}
$F_{1211}$(none) = {none}
$F_{1212}$(pitch) = {neg, pos}
$F_{1213}$(yaw) = {neg, pos}
$F_{1214}$(roll) = {neg, pos}
$F_{11}$(aah) = {on, off}

### A. Desirable Properties

For the functional ontology to be verifiable (complete, consistent, and correct), there are a number of properties that are desirable, some resulting from the functional structure and definition.

(1) Each object is defined (otherwise the ontology is not complete).
(2) Each object has a finite set of unique functional relationships (otherwise the system is not well defined).
(3) Each functional relationship is defined in the ontology (otherwise the ontology is not complete).
(4) Each object has a unique representation (otherwise there will be redundancy in the ontology). This property is guaranteed by the representation.

### B. Discussion

Variable naming approaches are not unique. As a result, the resultant set of trees also is not unique. In the case of the above example, an alternative set of names could be developed as follows in what is referred to as NAME SET 2. In this set, the number of objects increases by one and the number of functional relationships decreases by one.

*Objects*
$F_1$(gyroscope) = gyro
$F_1$(gyroscope movement) = gyro-movement
$F_1$(automatic attitude hold) = *aah*

*Functional Relationships*
$F_{11}$(gyro) = {on, off}
$F_{11}$(gyro-movement) = {none, pitch, roll, yaw}

$F_{111}$(none) = {none}
$F_{112}$(pitch) = {neg, pos}
$F_{113}$(yaw) = {neg, pos}
$F_{114}$(roll) = {neg, pos}
$F_{11}$(aah) = {on, off}

Alternatively, the names could be represented as in NAME SET 3, which is the most parsimonious representation of the three. However, it is also the only one of the three where for some set ($F_{11}$(gyro) = {on, off, movement}), some, but not all values have other branches in the tree.

*Objects*
$F_1$(gyroscope) = gyro
$F_1$(automatic attitude hold) = *aah*

*Functional Relationships*
$F_{11}$(gyro) = {on, off, movement}
$F_{111}$(movement) = {none, pitch, roll, yaw}
$F_{1111}$(none) = {none}
$F_{1112}$(pitch) = {neg, pos}
$F_{1113}$(yaw) = {neg, pos}
$F_{1114}$(roll) = {neg, pos}
$F_{11}$(aah) = {on, off}

Alternatively, rather than components of a single variable, these components could be assembled as two or more variables. For example, "gyro movement" could be a variable independent of the types of movement, e.g., "pitch neg." This wide range of representations suggests that whichever is chosen will have some structure that can be exploited to ensure system quality.

## V. ONTOLOGY SPECIFICATION OF INDIVIDUAL VARIABLES

A variable will defined as an $n$ tuple of objects and functional relationships, for $n \geq 1$ where there is a tuple for each object and related function. Tuples form the variables of interest that are used in the system, where each variable is a tuple formed from the related sets of ordered variables resulting from all paths from the root to the end of each branch in the tree. So,

$$F_1(\alpha), F_{1j}(F_1(\alpha)), \ldots$$

*Example*. Using the variable components described in the previous section, there are three different types of tuples that are defined, to ultimately generate the variable names. For NAME SET 1, we have

$F_1(gyroscope) = gyro$, $F_{11}(gyro)$ yields the two variables

gyro on
gyro off

*aah*, $F_{11}(aah)$
(aah off)
(aah on)

*gyro*, $F_{12}(gyro)$, $F_{121}(movement)$, $\{F_{1211}(none)$, $F_{1212}(pitch)$, $F_{1213}(yaw)$, $F_{1214}(roll)\}$
(gyro movement none none)
(gyro movement pitch neg)
(gyro movement pitch pos)
(gyro movement roll neg)
(gyro movement roll pos)
(gyro movement yaw neg)
(gyro movement yaw pos)

### A. Desired Properties

For the functional ontology to be verifiable (complete, consistent, and correct) there are a number of properties that are desirable.

(1) Each variable in the ontology is unique (otherwise there is redundancy)
(2) Each variable tuple defined by object and functional relationships is included in the ontology (otherwise the ontology is not complete).

## VI. ONTOLOGY SPECIFICATION OF THE ROLE OF VARIABLES

Variables can have different roles in the ontology. One category of existent information is whether or not a variable is an independent or a dependent variable. Let $A$ be the set of independent variables and let $Z$ be the set of dependent variables. For all $\alpha$, each variable represented as a tuple, $(F_1(\alpha)$, $F_{1j}(F_1(\alpha)), \ldots) \in$ either $A$ or $Z$, but not both. In terms of rule-based systems, this roughly maps into conditions and consequences, respectively. To the extent that those roles can be specified as unique occurrences, the ontology provides additional structure. Additional categories can also be generated to reflect other types of categories of variables, if they exist.

*Example*. One set of definitions for categories would put "gyro on" as an independent variable and gyro movement... as a variable dependent on gyro on. However, if we assume that gyro movement and movement type (e.g., pitch neg) are treated as variables, then movement type is dependent on gyro movement. In addition, we see that gyro movement and gyro on are parallel structures.

### A. Desirable Property

Ontology structures of roles of variables have a number of desirable characteristics that facilitate verification and validation.

(1) Variable roles are can be uniquely specified for all variables, given a particular set of variable components.

## VII. ONTOLOGY SPECIFICATION OF SETS OF VARIABLES

Using individual variables and their type categorization, tuple relationships between variables can be specified as sets of variables. For example, sets of independent variables could appear together or sets of independent and dependent variables could appear together as part of the causation phenomena being modeled by the system. In the case of a rule-based system, this would translate to establishing categories for sets of variables, such as feasible sets of variables that can be independent variables together, or sets of variables that can be dependent variables together or variables that can be dependent–independent sets. Such knowledge could emerge as understanding increases through system development.

Further, in some settings, relationships between sets of variables can be established as part of the ontology. For example, McBride and O'Leary[11] used the Cartesian product ($\otimes$) to relate all pairs of certain sets of variables. The operator $\otimes$ defines all pairs between specified variables, e.g., $(F_1(\alpha), F_{1j}(F_1(\alpha)), \ldots) \otimes (F_1(\beta), F_{1k}(F_1(\beta)), \ldots)$. The advantage of being able to specify Cartesian product relationships is that systems can be made to easily generate those relationships, rather than have humans intervene to generate them.

*Example*. Continuing with the above example, pairs of variables can be specified. The $\otimes$ operator is illustrated in the following examples of variable pairs. Then,

$[aah,\ F_{11}(aah)] \otimes [gyro,\ F_{12}(gyro),\ F_{121}(movement),\ \{F_{1212}(pitch),\ F_{1213}(yaw),\ F_{1214}(roll)\}]$

(*aah* on) (gyro movement pitch neg)
(*aah* on) (gyro movement pitch pos)
(*aah* on) (gyro movement roll neg)
(*aah* on) (gyro movement roll pos)
(*aah* on) (gyro movement yaw neg)
(*aah* on) (gyro movement yaw pos)

$[gyro,\ F_{12}(gyro),\ F_{121}(movement),\ \{F_{1211}(none),\ F_{1212}(pitch),\ F_{1213}(yaw),\ F_{1214}(roll)\}]$ (*gyro on*)

(gyro movement none none) (gyro on)
(gyro movement pitch neg) (gyro on)
(gyro movement pitch pos) (gyro on)
(gyro movement roll neg) (gyro on)
(gyro movement roll pos) (gyro on)
(gyro movement yaw neg) (gyro on)
(gyro movement yaw pos) (gyro on)

In addition, these sets of variables can also exploit whether they are drawn from a dependent or an independent variable set or one from each.

### A. Desired Properties

To assure verifiability, there are a number of desirable properties.

(1) Each feasible $n$ tuple is unique (otherwise there is redundancy).
(2) Each feasible $n$ tuple is included (otherwise the ontology is not complete).
(3) Each $n$ tuple generated is feasible (otherwise, ontology is not correct).
(4) Each Cartesian product can be specified at the definition level without having to elicit each pair (otherwise there is an opportunity for an error).

Ontologies can be used to specify a wide range of relationships, including specification of $n$ tuples of variables (not just $n$ tuples of components that make up variables), symmetric relationships between $n$ tuples, and/or other types of functional relationships.

## VIII. ONTOLOGY SPECIFICATION OF CARDINALITY

Let $|\alpha|$ indicate the cardinality of $\alpha$, where cardinality refers to the number of something. Cardinality relationships can also be specified as part of the ontology definition. Cardinality naturally appears as a part of many real world phenomena. Ontology specification of cardinality can relate to specification of the number of times an object, a variable pair, or any other aspect should appear in a system.

*Example*. Continuing the example from above, the sample system has certain emergent specifiable cardinalities.

| *Cardinality* | *Pair* |
|---|---|
| 49 | (gyro movement none none) (gyro on) |
| 4 | (gyro movement pitch neg) (gyro on) |
| 4 | (gyro movement pitch pos) (gyro on) |
| 4 | (gyro movement roll neg) (gyro on) |
| 4 | (gyro movement roll pos) (gyro on) |
| 4 | (gyro movement yaw neg) (gyro on) |
| 4 | (gyro movement yaw pos) (gyro on) |

However, different naming conventions and different variable sets can result in different cardinalities. For example, as noted above, we might treat gyro movement as a separate variable and the type of movement (e.g., pitch neg) also as a variable. In this case, there are different sets of variable cardinalities.

| | |
|---|---|
| 49 | (gyro movement) (none none) (gyro on) |
| 4 | (gyro movement) (pitch neg) (gyro on) |
| 4 | (gyro movement) (pitch pos) (gyro on) |
| 4 | (gyro movement) (roll neg) (gyro on) |
| 4 | (gyro movement) (roll pos) (gyro on) |
| 4 | (gyro movement) (yaw neg) (gyro on) |
| 4 | (gyro movement) (yaw pos) (gyro on) |

In addition, we can derive additional cardinality pairs, such as

| | |
|---|---|
| 73 | (gyro movement) (gyro on). |

This later cardinality measure is a looser constraint than the detailed set of 49, 4, etc.

### A. Desirable Property

(1) Cardinality can be specified for all variables and $n$ tuples of variables.

## IX. EMERGENT KNOWLEDGE

With different NAME SETS and when different variables are used to model the same phenomena, then there is different emergent behavior. When the variables "(gyro movement)," "(pitch neg)," "(gyro on)," or "(gyro movement pitch neg)" "(gyro on)" are used, there are differences in what variables are independent or dependent variables, what variables appear as tuples, what are the cardinalities of tuples, etc.

### A. What is Emergent Knowledge?

When a variable representation is generated and implemented, there are certain characteristics that emerge because of the choice of the name and what is included in that name, this is emergent knowledge. That emergent knowledge can be used for facilitating system quality.

Emergent knowledge does not depend on the form of knowledge representation being used. This article argues that emergent knowledge should be included in ontology specifications. Its inclusion can link system specification through ontology with what has been seen primarily as one of the final tasks, verification and validation.

In general, emergent knowledge is knowledge about the variables that can constrain the behavior of the system, but that ties directly to the concepts in the ontology. However, if the knowledge representation is well established then

emergent knowledge can include constraints resulting from implementation of the variables in the particular knowledge representation.

### B. Advantage and Disadvantage of Emergent Knowledge

The primary advantage of using emergent knowledge is that it allows development of a tighter ontology specification, with more structure and constraints. That structure and those constraints can facilitate development of systems that are consistent, complete, and correct. Perhaps the primary disadvantage of using emergent knowledge is that it can make it more difficult to make changes to the system. Adding new knowledge to the system likely would change the specification of the ontology, forcing changes in both the system and the ontology. However, if the knowledge did change then the ontology would likely change also.

### C. What Kinds of Emergent Knowledge Probably Should Not be Used?

Certain types of emergent knowledge probably should not be embedded in the ontology. First, knowledge representation—specific knowledge probably should not be embedded in the ontology. Such knowledge would limit re-use to similar knowledge representation structures. Second, emergent knowledge that changes frequently or is likely to change should not be used in the ontology.

### D. How Much Emergent Knowledge Should be Used in the Ontology?

An ontology can be overspecified using too much emergent knowledge or underspecified, using too little emergent knowledge. How much emergent knowledge that should be used depends on the developer and the developer's commitment to the ontology and need for flexibility at the system level. In any case, cost benefit analysis can be used to decide which structures provide the greatest benefit.

## X. SUMMARY

This article differentiated between emergent and existent types of knowledge for use in ontologies. Emergent knowledge is dependent on the implementation of naming variables and what concepts are made into different variables. Emergent knowledge can include the role of the variable, cardinality of the variable, appearance of the variable with other variables, and a range of other knowledge. Emergent knowledge can be used to establish expectations about a system from its ontology, constraining the system. Accordingly, emergent knowledge can be used to improve system quality through facilitation of identification of anomalies. Building emergent knowledge into ontologies provides a means to shift what has been traditionally testing activity into what is commonly viewed as design.

## References


1. O'Leary D. Artifacts: Toward a theory of verification and validation. Int J Intell Syst 1994;9(9):853–866.
2. Gruber T. A translational approach to portable ontologies. Knowledge Acquisition 1993;5(2):199–220.
3. Cutkosky M, Engelmore R, Fikes R, Genesereth M, Gruber T, Mark W, Tenenbaum J, Weber J. PACT: An experiment in integrating concurrent engineering systems. Computer 1993;January:28–37.
4. Waterman A, Preece A. Knowledge reuse and knowledge validation. In Verification and Validation of Knowledge-based Systems—Papers from the 1997 AAAI Workshop, p 33–39.
5. Gomez-Perez A. Some ideas and examples to evaluate ontologies. In Proceedings of the Eleventh Conference on Artificial Intelligence, Los Angeles, CA, February 1995, p 299–305.
6. Adrion W, Branstad M, Cherniavsky J. Validation, verification and testing of computer software. ACM Comput Surveys 1983;14(2):159–192.
7. Gruber T, Bibliographic data ontology, 1994. ftp:// hpp.stanford.edu/ pub/ knowledge-sharing/ ontologies/ html/ bibliographic-data/.
8. Preece A, Shinghal R, Batarekh A. Verifying expert systems: A logical framework and a practical tool. Expert Syst Appl 1992;5:421–436.
9. Lawler D, Williams L. MMU FDIR automation task. Final Report, Contract NAS9-17650, Task Order EC87044, Houston, TX, McDonnell–Douglas, Astronautics Company, 1988.
10. Landauer C. Correctness principles for rule-based expert systems. Expert Syst Appl 1990;1(3):290–316.
11. McBride R, O'Leary D. A generalized network modeling system for scheduling. Ann Oper Res 1997;75:355–372.