Mary Parmelee. Design For Change: Ontology-Driven Knowledgebase Applications For Dynamic Biological Domains. A Master's Paper for the M.S. in I.S. degree. November, 2002. 107 pages. Advisor: Brad Hemminger

Post-genomic biology is producing a plethora of rapidly changing complex data. Yet extracting useful information from this data is limited by current knowledge management methodologies. Biological knowledge management is complicated by ambiguous nomenclature, cultural differences between biologists and computer scientists, and conventional database technology that was not designed to support rapidly changing complex domains. A recent trend in ontology-driven database design has emerged to address this challenge. While ontologies provide effective knowledge models, attempts to transform ontologies into knowledgebases have revealed an impedance mismatch or ontology transformation gap. A unique methodology called Ultra-Structure Theory (UT) may provide an ontology transform solution that supports large scale, dynamic biological domains by expressing the complexity in data rather than programming code. This thesis aims to survey ontology and database theory and methodologies, and describe how UT integrates and extends them to provide a flexible, semantically expressive knowledgebase solution using standard relational database technology.

Headings:

Application software – Development

Information storage and retrieval--Design

Information systems—Design

Knowledge representation

Ontology

DESIGN FOR CHANGE: ONTOLOGY–DRIVEN KNOWLEDGEBASE APPLICATIONS FOR DYNAMIC BIOLOGICAL DOMAINS

by
Mary C. Parmelee

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

November, 2002

Approved by:

____

Advisor

Table of Contents

1. INTRODUCTION

The emergence of post-genomic biological disciplines, high throughput data acquisition methods, and large-scale science projects are rapidly generating a plethora of complex, biological data. While the advent of disruptive technologies such as such as mass spectrometry and microarray technology will eventually lead to rapid knowledge discovery and new biological paradigms, current knowledge management methodologies are a limiting factor. However, clinically significant discovery remains dependent upon scientists' ability to access the data, understand it, make statistical inferences based on it, and share it among the scientific community.

One of the major challenges of post-genomic biological research is to develop methodologies that manage this data in a context that enables the scientist to translate raw data into workable models and viable hypotheses. Due to the volume and complexity of the data, computers are required for data processing and storage. Although the use of computers effectively addresses information bottleneck issues that are created by volume, the complexity issue is exacerbated by a legacy of ambiguous genetic nomenclature, a cultural gap between biologists and computer scientists, rapidly changing functional requirements, and an impedance mismatch between knowledge representation and database technologies.

Since databases are the storage medium for biological data, devising a framework for organizing and manipulating complex biological data is inherently dependent upon the optimization of database design and implementation for knowledge management. Data models and the databases that are derived from them will need to reflect intricate semantic relationships between millions of biological entities. These models will be subject to frequent change and even radical transformation as experimental results deepen our understanding of large biological networks. Standard information system development processes are not designed to address these issues.

A recent trend in ontology-based design methods has emerged as a means of providing a logical semantic framework for organizing biological data. The intentional use of ontologies is twofold: to make them available to the biological community as a semantic resource in hope of developing defacto standards for knowledge representation, and to use them as a basis for database design. Early nomenclature management projects have developed into knowledge leveraging initiatives, that are developing large ontology-driven knowledgebases. Projects such as Ecocyc and The Gene Ontology have recently made great strides in capturing biological knowledge including complex processes of simple biological organisms.

While ontologies provide an effective framework for representing biological knowledge on an abstract level, efforts to map ontologies to database structures have revealed an impedance mismatch or ontology transformation gap. Extracting knowledge from complex biological systems requires the enforcement of a myriad of semantic constraints or rules that interact to form a body of knowledge. The ontology transformation gap refers to the incompatibility or mismatch between ontology and database knowledge representation systems. Without the development of extensive programmatic augmentation, current database modeling techniques and methodologies are too rigid to represent the multidimensional semantic constraints and syntactic dependencies that are required for accurate knowledge representation. Even more flexible object oriented databases have significant limitations.

Database design approaches that use existing methodologies are likely to hit a complexity barrier at a fairly low level of usability and are not conducive to the rapid change of the domain. Efforts to map ontologies directly within the confines of a standard relational design require complex queries or extensive programming code in order to maintain semantic consistency; and progress in developing new knowledge representation tools have met with only limited success. This approach often results in suites of specialized tools and new query languages that are expensive to develop, difficult to learn, and face constant reprogramming in order to keep pace with the rapidly changing domain.

This thesis aims to describe a unique ontology–driven approach to database design known as Ultra-Structure Theory (UT). UT integrates current theoretical and methodological perspectives, and extends them to provide a practical and logically valid methodology that bridges the ontology transformation gap and accommodates perpetual change.

This thesis will provide a foundation for analysis by: outlining the issues affecting biological knowledgebase design, describing ontological and database theory and methodologies, reviewing ontology-based biological database methodologies, and describing UT principles and applications. Finally, I will provide a comparative analysis of the advantages of UT over existing methodologies, followed by a discussion of potential challenges, and future research.

2. BACKGROUND AND SIGNIFICANCE

Designing a repository that represents or simulates dynamic complex biological knowledge poses a multifaceted challenge. This section will outline the dimensions of the problem, and introduce the benefits of an Ultra-Structure approach.

2.1 The Critical Bottleneck

Although there has been a great deal of progress in the field of biological knowledge management, the rate of production and increased complexity of biological data continues to outpace current knowledge management capabilities. For example, GenBank, the largest of the genomic databases, now accommodates $>10^{10}$ nucleotides of nucleic acid sequence data and doubles in size every year (Roos, 2001). Since its inception in 1992 Genbank has stored its data in flat files, which are the relational equivalent of storing everything in one table. There is little structure and no mechanism for representing semantic relationships. What's more, acquiring data from flat files requires a specialized computer program that reads the file before the data can be downloaded and must be updated if the file format changes.

The disparity between biological data production and useful data access and analysis is likely to increase downstream in the absence of effective knowledge management and transformation techniques. Research organizations who develop computer systems that capture and apply the knowledge of domain experts, and make it accessible in a flexible and cost effective manner will have a competitive advantage.

2.2 The Cultural Divide

The recent blending of the disciplines of computer science (CS) and biology has not been without friction. Stanford molecular geneticist David Botstein, referred to it as the "two culture" problem. Botstein theorizes that biologist's perception of open, interesting problems might not appeal to computer scientists or mathematicians. Moreover, since each discipline has a highly technical and complex jargon, effective communication of problems, concepts and ideas is troublesome. Few scientists have expert knowledge of both computer science and biological research. However, an increasing number of computer science projects will be driven by biological problems. These problems will likely not be pursued by biologists (Fujimura, 1995).

2.3 The Nomenclature Problem

> "Biologists would rather share their toothbrush than share a gene name," says Michael Ashburner, joint head of the European Bioinformatics Institute (EBI) at Hinxton near Cambridge, and one of GO's founders. "Gene nomenclature is beyond redemption." (Pearson, 2001)

Nomenclature problems in human genetics were first recognized in the 1960's. By 1979, guidelines for human gene nomenclature were presented at the Edinburgh Human Genome Meeting (HGNC, 2001). However, resistance by the scientific community to conform to nomenclature standards and the recent discovery that closely related genes often exist in different organisms has significantly escalated the terminological discord (Pearson, 2001).

The severity of the problem was recently documented when a research team in Oslo Norway developed computer software to search for biological associations between genes based on their co-occurrence in published paper abstracts. Testing the software involved scanning over 10 million records in the Medline database. The scan identified 22,008 distinct human genes of which 10,352 had aliases. For example, the gene, *SELL*, or *selectin L.* had 15 aliases. Contributing to the problem is over 4,000 gene abbreviations that refer to multiple genes, and numerous gene homographs between different species. One example is the yeast homologue of the human gene *PMS1*, which is named *PMS2*; while yeast *PMS1* corresponds to human *PMS2* (Pearson, 2001).

Two international nomenclature workshops, held in 1997 (Blake) and 1999 (White), concluded that due to the severity of the problem, the only cross-species attempt to standardize gene names would be for mammals.

2.4 The Complexity Problem

Emerging, data-rich biological domains such as structural and functional proteomics epitomize rapidly changing complex systems. Rapid change of a complex information domain evokes new perspectives or world views that attempt to explain domain dynamics. Complex systems theory takes a holistic approach in analyzing complex systems. This theory exemplifies the Gestaltian phrase, "The whole is more than the sum of its parts". In this world view, systemic outcomes represent the simultaneous interactive properties and behaviors of all parts of a complex system. These behaviors and properties are governed by similar universal principles.

All complex systems have universal properties. Considering these global properties of complex systems, can lead to the comprehension of system specifics in the context of their commonalities. Identifying the universal properties of a system, characterizes the system at a high level of abstraction. These universal simplifications can impose a system structure that enables observers to perceive a system's integrated network of perceptual patterns. Articulation of these principles enable us to approach the study of particular systems with systematic guidance (Bar Yam, 2002).

In his recent publication , *"Unifying Principles in Complex Systems",* Yaneer Bar-Yam of the New England Complex Systems Institute in Cambridge, MA, outlines the goals of complex systems research (2002).

- Understand the development and mechanisms of patterns of behavior
- Understand the unifying principles of organization

- Understand what is universal and what is not
- What are the classes of universal behavior and the boundaries between them
- What are the relevant parameters for description or for affecting
  the behavior of the system
- Develop the ability to capture and represent specific systems, rather than just accumulate data about them. In this context: to describe relationships, know key behaviors, recognize relevance of properties to function, and to simulate dynamics and response.
- Achieve a major educational shift toward unified understanding of systems, and patterns of system behavior.

These goals are applicable to any complex system including computer systems. Recognizing the unifying principles of biological systems as well as that of the computer systems in which we represent them, will produce patterns of function that have a much higher resilience to failure and error and a higher adaptability to changing conditions and rules. Many tools designed for specific domains can be adapted for more general use by recognizing their universal applicability (Taylor, 2001).

2.5 The Current World View of Computer Systems

2.5.1 Systems engineering

Current systems engineering methodologies use a set of predetermined functional requirements as a roadmap for information system development. Defining functional requirements can be characterized as defining a snapshot of the information landscape or worldview from the end user's perspective. Figure 1 represents a typical pattern of current systems development. This method works well for relatively stable information domains. However, functional requirements are more descriptive than prescriptive and thus are dependent upon changes in the end user's state of knowledge in the context of observed changes in the information domain itself.

Although recent user-centric techniques such as "Contextual Design" (Beyer and Holtzblatt, 1999) and "Pragmatic Product Development" (2002) have improved the development process of functional requirements, and rapid prototyping is enabling developers to accommodate changing functional requirements during system development, there is currently no design methodology that focuses on optimizing information systems to accommodate rapid, continuous change (Long and Denning, 1995). Figures 2 (InContext, 2002) and 3 (Pragmatic Marketing.com, 2002) diagram the "Contextual Design" and "Pragmatic Product Development" methods. Even advanced requirements development that uses predictive analytics for determining functional requirements such as the "Decision Cycle Method" are based on design time optimization.

A relatively new and dynamic biological domain such as genomics is difficult if not impossible to predict. There will likely be numerous unexpected results as experimental inferences are elucidated. The well publicized overestimate of the number of genes comprising the human genome is one example of the potential for inaccuracy in scientific predictions. Many biological disciplines are changing so rapidly that modeling systems based on static functional requirements is no longer an effective methodology for defining system requirements.

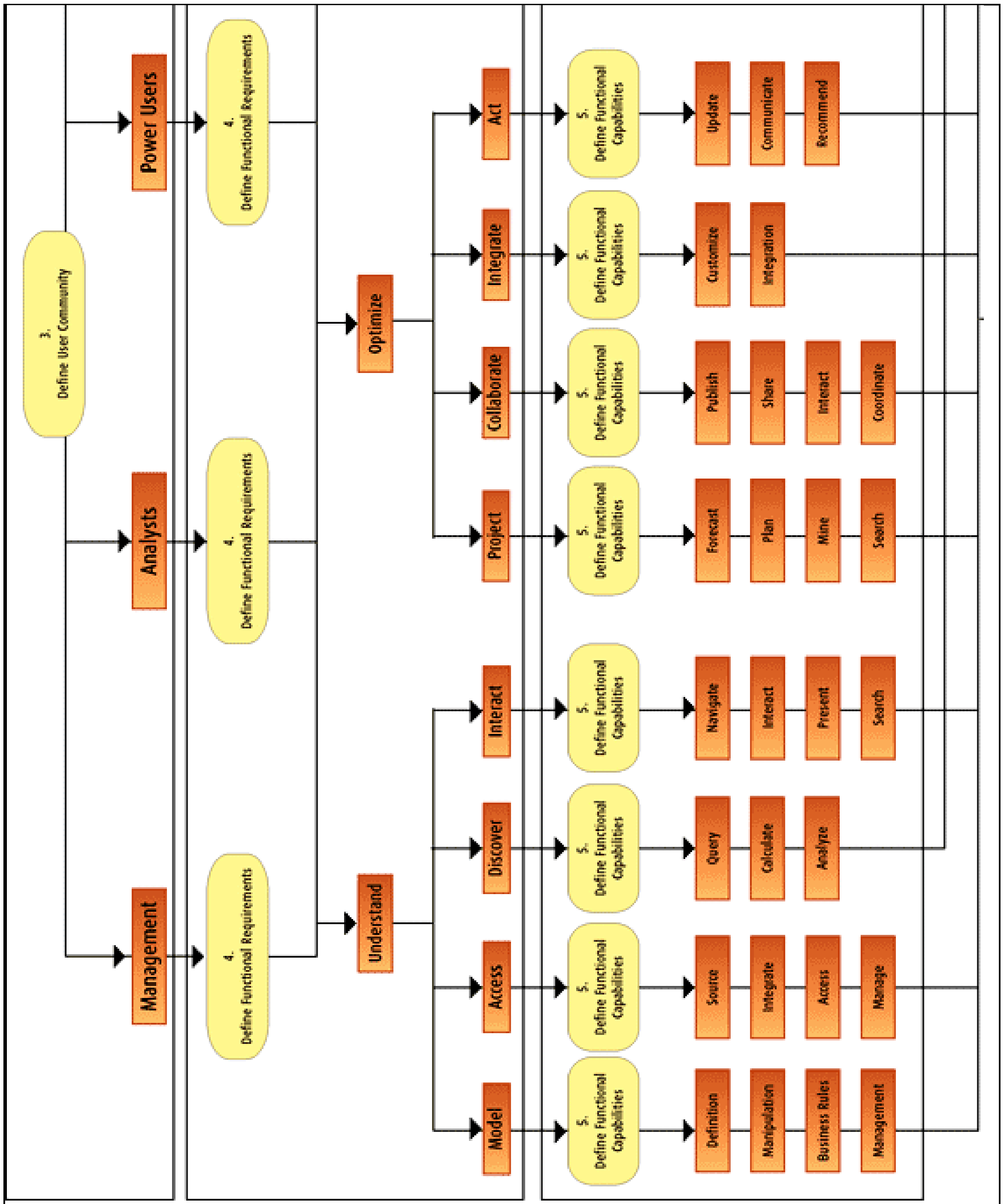


Figure 1: Typical software development life cycle

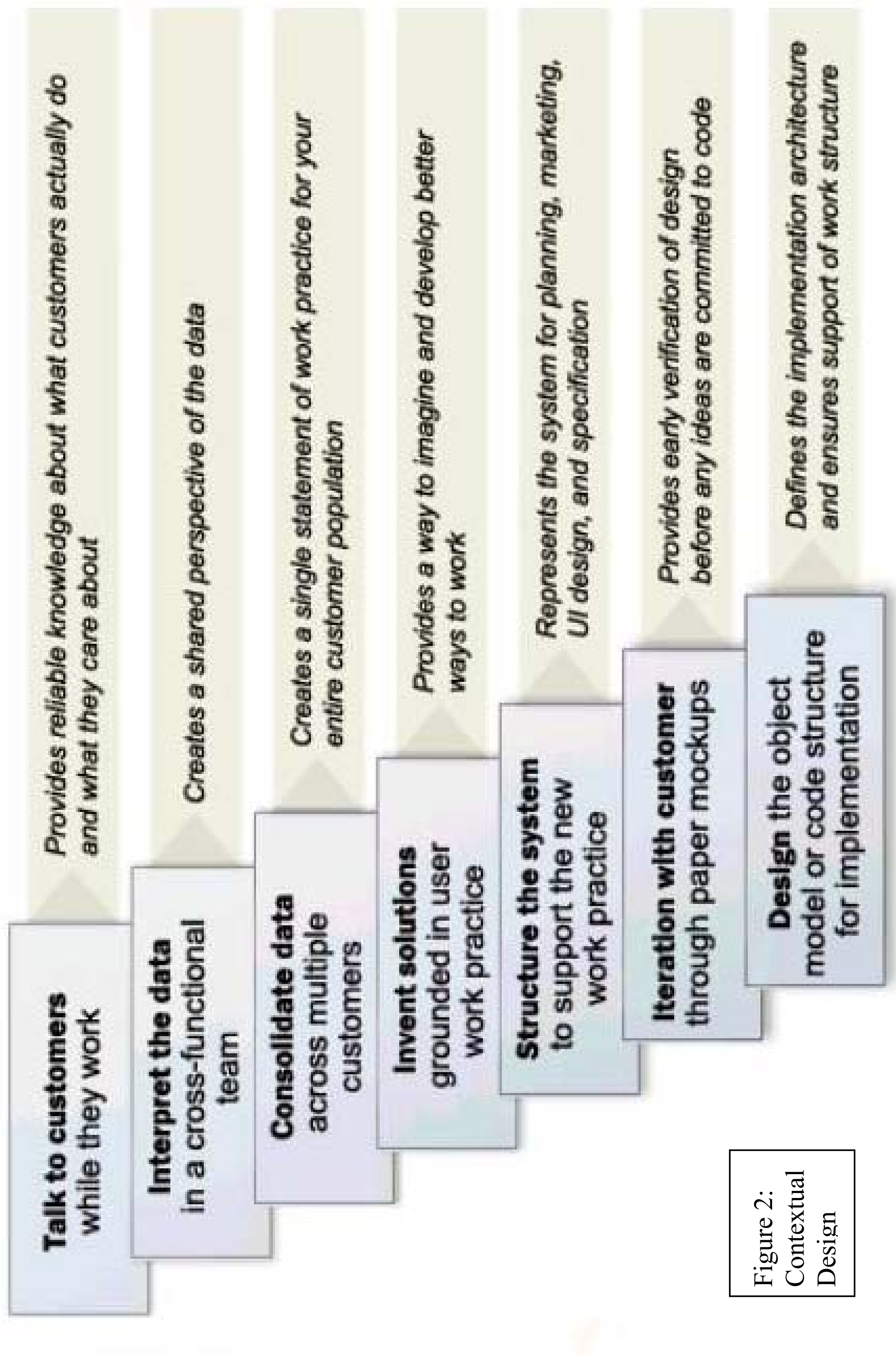

Talk to customers while they work
Provides reliable knowledge about what customers actually do and what they care about
Interpret the data in a cross-functional team
Creates a shared perspective of the data
Consolidate data across multiple customers
Creates a single statement of work practice for your entire customer population
Invent solutions grounded in user work practice
Provides a way to imagine and develop better ways to work
Structure the system to support the new work practice
Represents the system for planning, marketing, UI design, and specification
Iteration with customer through paper mockups
Provides early verification of design before any ideas are committed to code
Design the object model or code structure for implementation
Defines the implementation architecture and ensures support of work structure

Figure 2: Contextual Design

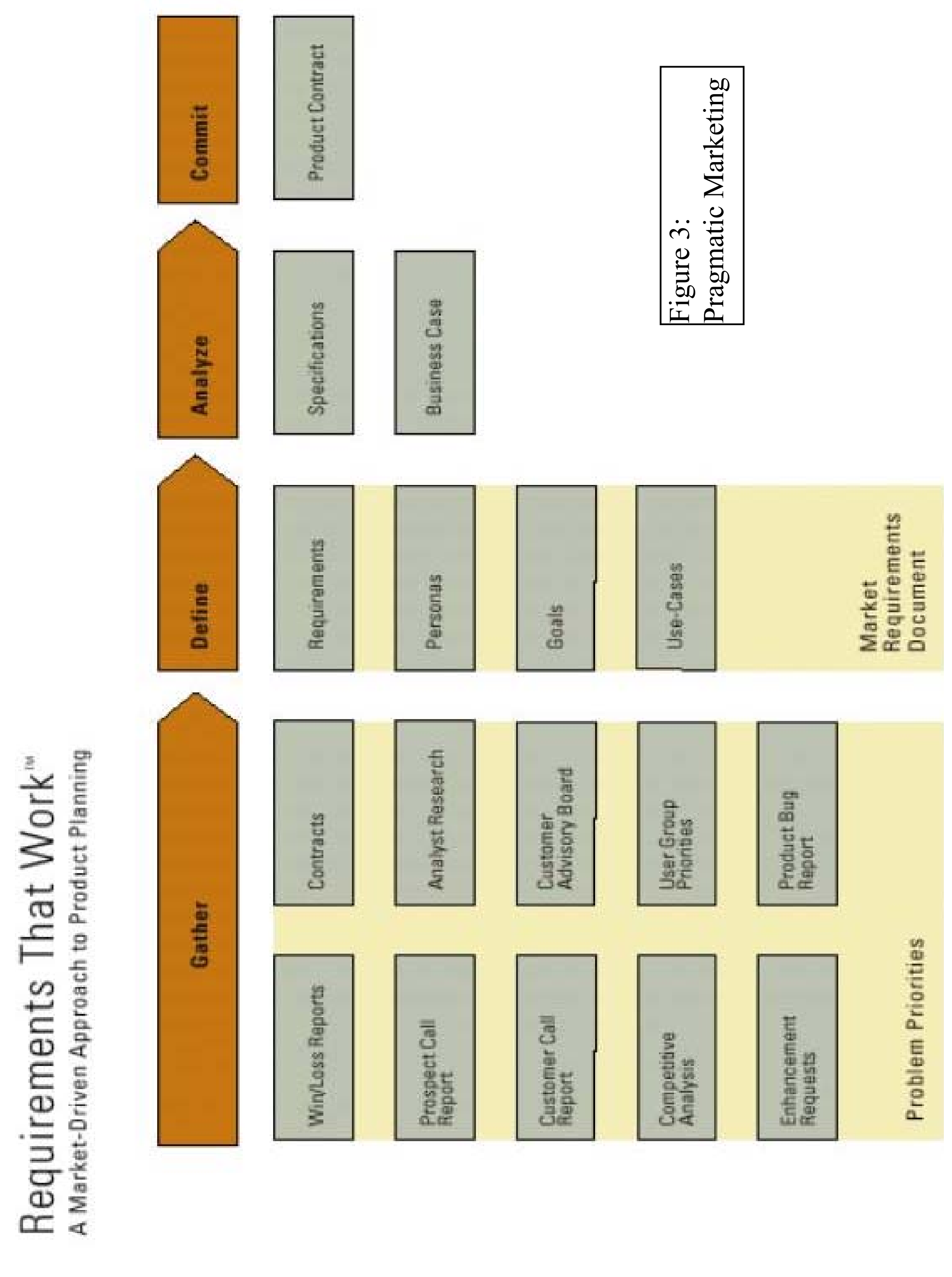


Figure 3: Pragmatic Marketing

2.5.2 Database applications

All computer data is stored in a digital (0 or 1) binary format. The current world view of digitally stored data recognizes computer programs as a special kind of data that contains encoded instructions for the manipulation of application or end user data (Whatis, 2002). Computer programs are written in a programming language and translated to binary format by a compiler or assembler.

Data is human readable information that is typically written in natural language by an end user and then converted to binary format. Data represent the facts that are accessed, managed, and updated by programs, they provide the parameters within which programs must act upon the data. All actions or procedures of an application are controlled by programs that are written based on a static world view at the time the program was created. End-users are able to modify only data or the facts that the software manipulates, not the actions or procedures that control the data (Long and Denning, 1995).

Databases are computer applications that manage collections of data. As with all applications, the algorithms that define and manipulate the data are represented in programs such as queries and stored procedures. Therefore, changes or manipulation of the data typically involves adding, deleting or modifying software code. Changing code requires knowledge of one or more programming languages and frequent changes can be expensive and time consuming (Long and Denning, 1995).

2.5.3 Knowledge based systems

There is no standard definition of what constitutes a knowledgebase. Typically a knowledgebase is composed of a database of domain specific information that is optimized for knowledge extraction (Whatis, 2002). Optimization is achieved by expressing the facts and rules of the domain in a knowledge representation language. This expression often takes the form of an ontology, a concrete specification of the abstract concepts that define a domain (FOLDOC, 2002). Inference engines are software algorithms that use the rules of logic governing knowledgebases to infer or learn new facts by deductive reasoning. Knowledgebases and the inference engines that manipulate them are called knowledge based systems.

2.5.4 Expert systems

While knowledge based systems in general are designed to capture and extract knowledge from a knowledgebase, expert systems use a similar rule-based approach to simulate the knowledge of a domain expert. Knowledge is collected in the form of a rule base. A rule base is the set of all possible rules that are applicable to a given domain. These rules generally take an "If *fact*→Then *conclusion*" deductive format. Multiple *facts* can be strung together to reach more specific conclusions. The expert system inference engine processes the rules and generates new rules and facts from existing rules and facts by finding rules that will fire based on given or known facts.

Typically, an expert system will have an interface that asks the user a series of questions. The answers become the "seeds" or factors from which the inference engine bases it's conclusion. The inference engine then traverses the decision tree, comparing these facts to every rule in the system. It reaches a conclusion based on which rules match the given facts. Figure 4 is a simplified diagram of an expert system that identifies birds (Beck, 2002).

An example of a rule for an eagle might be:

IF Large **AND** Flies **AND** Has a crest **AND** is Bald →**THEN** Is an Eagle.

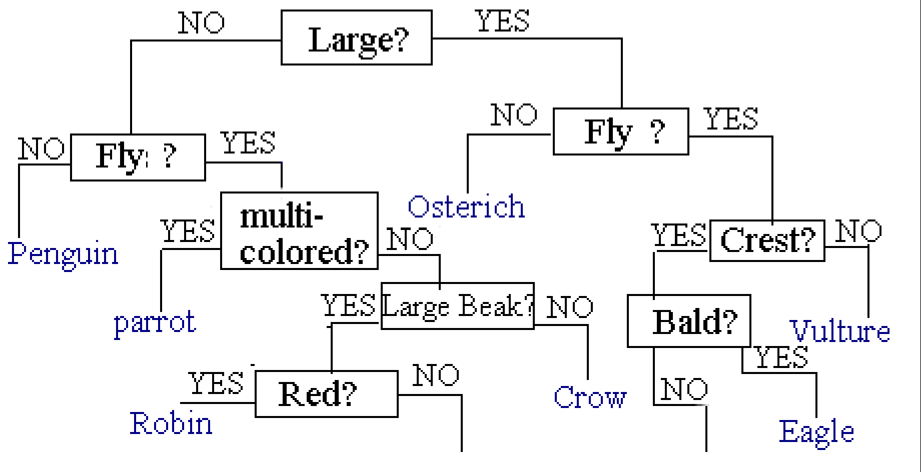


Figure 4: Expert system model

Expert systems have high speed, high consistency, and perfect attention to detail. (Long and Denning, 1995). Rule bases can contain over a thousand rules, however rules are not equal to mathematical proofs. They are closer to heuristic rules of thumb. Due to the heuristic nature of expert systems, their accuracy can decline significantly as the number of rules increase or the domains broadens. Hence, expert systems do not scale well for broad complex biological systems.

2.6 The Ontology Transformation Gap

Ontologies capture the semantic knowledge of complex systems via logical abstraction. They characterize the focus or aboutness of a domain by mapping the set of representational concepts and the range of interrelations between concepts to form the domain network as a whole. Their logical principles overlap considerably with complex and expert systems theory.

The intricate semantic network of ontologies is typically formalized in a complex ontology metalanguage such as DAML+ OIL (Darpa Agent Markup Language + Ontology Inference Layer). These knowledge representation languages are semantically expressive and syntactically flexible enough to provide a contextual framework that preserves the semantic linkages of an ontology in a machine readable syntax. Ontology metalanguages are intended for use by sophisticated inference engines or agents that will crawl the proposed Semantic Web (Future WWW) and perform complex queries. However, these engines do not yet exist in their proposed form.

Recently ontology–driven database design has gained tremendous popularity in the biological community as a method of addressing the nomenclature and complexity issues of the discipline. Although many long term, large scale ontology driven database projects are underway, no clear methodology for mapping ontologies to database schemas has been developed. Transforming biological ontologies to database schemas requires the ability to express the syntactics of the schema based on the semantic constraints imposed by the ontology (Biskup, 1998). Semantics and syntactics represent complementary aspects of language, that together provide meaning in context.

The ontological transformation gap describes an impedance mismatch between current ontology and database schemas. This gap makes it difficult to translate ontologies within the constructs of standard models such as Entity-Relation diagrams, and even more difficult to translate into standard relational schemas. The structure of the relational schema requires the use of complex queries written in a DDL (data definition programming language) to capture most contextual semantic representations. Object models offer more flexibility and support for complex data types that enable more expressive semantic representations, but they are slow to retrieve simple data types, are more programatically intensive than relational database designs, and have scalability and support issues.

While biological ontologies have already contributed greatly to solving complexity and nomenclature problems, their benefits for database design are limited by this schematic mismatch, which can be ameliorated by an efficient transformation methodology.

2.7 Bridging the Gap While Providing for Change

While current information systems design methodologies are sufficient for modeling most types of systems, typically the rate of change of a system is considered only in the context of system maintenance. As biology and computer science continue to integrate and expand, the velocity and pervasiveness of change and complexity of the domain will increase dramatically. As a result, new systems design methodologies will emerge that blend features of current models and extend them to design for change. Ultra-Structure Theory is one such methodology.

3 ONTOLOGIES

> The first layer of the Semantic Web consists of ontologies and taxonomies ... A huge amount of this is being done very desperately in the realm of biotech, for the human genome and new drug development. (Tim Berners Lee, August 30, 2001 keynote at Software Development East in Boston.)

3.1 Ontological Theory

Applied ontology is a form of knowledge representation. Its principles were derived from philosophical ontology, which is the study of being or existence. Although general ontologies exist that are not grounded in a specific domain, they do not apply to the subject matter discussed in this thesis. Hence, I will concentrate this discussion on domain ontologies.

Ontologies represent the complex whole of a conceptual space or domain by specifying the sum of its parts or the set of entities that exist within that space, and the set of interactions or relationships that hold between those entities. Ontologies are typically guided by logical rules of inference, usually in the same first order logic, "If→Then" form that is used by expert systems. Yet, they can range from intuitive natural language ontologies with no formalized logic to formal ontologies expressed in explicitly stated axioms. In this section I will begin by defining ontologically related conceptual distinctions, followed by a brief outline of the logical principles and premises that are inherent in ontology metalanguages.

3.1.1 Conceptual distinctions

3.1.1.1 The semantic triad

The purpose of developing a domain ontology is to create a semantically based organizational schema or semiotic view of a domain. The goal is to provide a common semantic structure for exchanging information about that domain. For example, in order to apply the genetic information that is being catalogued in numerous databases worldwide, biologists need the ability to identify related genes across databases. This is currently a daunting task due to the lack of semantic interoperability among these resources. Nomenclature standardization is not feasible, and numeric classification by ID has met with only limited success (Pearson, 2001). Ontologies provide the rich semantics that enables biologists to makes sense of a complex information space. The semantic benefit of ontologies has often been diagrammed as a semantic triad (Figure 5) in which a symbol evokes a concept of a something real.

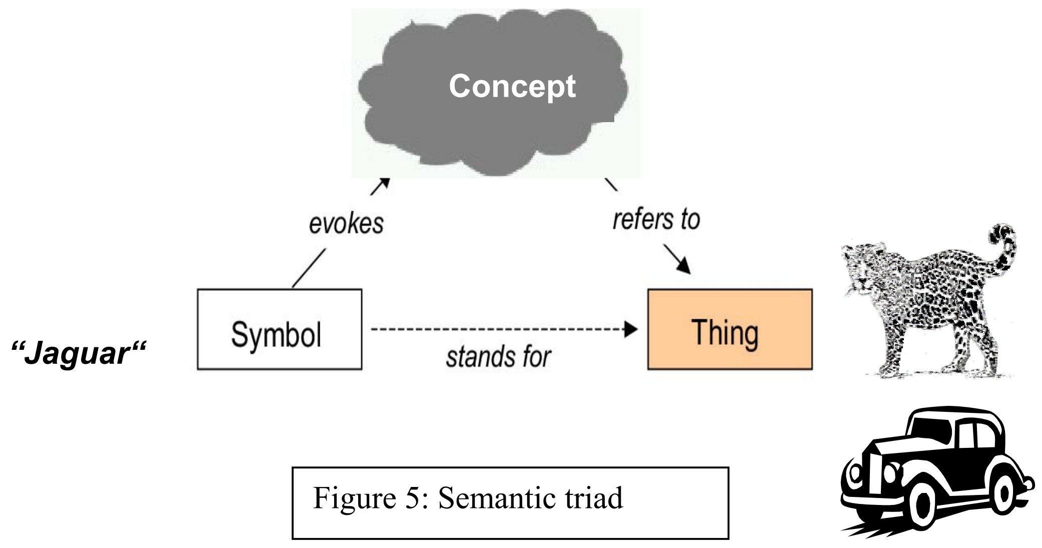


Figure 5: Semantic triad

3.1.1.2 Semiotics

Semiotics is the study of signs. Its two main branches are semantics and syntactics. Signs can be any symbol of expression that we associate with an object or a concept. In the development of ontologies the signs that we are typically concerned with are words. Semantics is the study of the conceptual meaning that we attach to signs or symbols. It is related to the function of a sign. The second branch of semiotics is syntactics, which is the arrangement or order of signs. When applied to language, syntax is referred to as grammar (Whatis, 2002). As I stated in the background section, semantics and syntactics are complementary properties of a language. Together they provide meaning in context. There is a positive relationship between semantic and syntactic flexibility. A syntax with a narrow range usually limits the range of semantic expressivity of that language (Stevens, 2000).

3.1.1.3 Sortal vs. non-sortal predicates

The notion of sortal vs. non-sortal predicates is important for distinguishing between predicates that are required for the definition of a concept (sortal) and more characteristic predicates (non-sortal) that describe aspects or attributes other than identifiers of a concept. Guarino defines a sortal predicate as:

- countable, e.g. the predicate allows a given object to be identified amongst other kinds of objects
- temporally stable, i.e. if the predicate holds for an object at a given time, it also holds for the same object at another time
- ontologically rigid, e.g. an object cannot lose a sortal property without losing its identity.

In contrast, non-sortal predicates, supply properties only for individuals already distinguished (Guarino, 1994). For example, compare the predicates 'red' and 'apple' to describe a concept of fruit. While the non-sortal predicate 'red' may be disassociated from the concept without affecting its identity, removing the sortal predicate 'apple' from the fruit concept causes it to lose its fruit 'type' identity (Guarino, 1994). The distinction of temporal stability is particularly important for conceptualizing a complex changing domain. This distinction is evident in the fact that an apple can be green before it is ripe and red afterward, but an apple is an apple for as long as it exists as a type of fruit. These principles provide a guideline for determining whether predicates should be defined as classes or properties of classes.

3.2 Knowledge Representation

Since applied ontologies are specifications by design, their logical principles are inherently woven into the framework of the knowledge representation languages or metalanguages in which they are expressed. Hence, I will describe the principles of ontology in the context of these language systems.

3.2.1 Frames

Frame-based ontologies take an object-oriented approach to characterizing ontologies that is similar to object oriented theory (OO). A frame is like an OO class that represents a class of objects or instances. Like OO the frames system has associated slots that are similar to OO attributes. The concepts in these slots may be frames themselves with their own set of attributes. Slots have type constraints such as 'kind-of' that allow for type-subtype subsumption inheritance. Frames also allow for the declaration of instances with the 'instance-of' slot. The semantic constraints of frames are described in the Open Knowledge Base Connectivy (OKBC) standard. Frames are intuitive and provide basic reasoning for the development of simple taxonomic hierarchies (Stevens, 2000).

3.2.2 Description logics

Rather than locking the ontology into a hierarchical tree structure, description logics (DL) generate taxonomies on the fly by describing but not defining primitive concepts and relationships then combining them to create complex concepts and relationships. For example the concepts of protein and reaction can be joined with the relationship catalyzes to define the composite concept of enzyme, a protein that catalyzes a reaction. The goal is to create ontology-based expert systems that allow an inference engine or reasoner to automatically generate new concepts based on the combinations of existing concepts thereby automatically generating logically consistent taxonomies. Description logics provide for richly expressive ontologies. The trade off is that more expressive languages require a more complex, processor intensive reasoner, which could be a scalability issue for complex systems (Stevens, 2000).

3.2.3 Hybrid systems

More recently metalanguages such as DAML+OIL have combined the benefits of frame based and description logic models. Table 1 is a summary of ontological definitions, and Tables 2 and 3 outlines the description logic as the set of constructors and axioms associated with DAML+OIL (Stevens, 2002).

Table 1: Ontological Concepts Summary (Stevens,2002)

CONCEPT	DESCRIPTION	EXAMPLE
Concepts	class, set, type, predicate	event , idea, object
Properties	concepts that are attributes of other concepts	weight, age, location
Taxonomy	hierarchy of terms representing concepts with inherited properties	Dog $\xrightarrow[\text{is a}]{}$ mammal
Relationship	action, process, role, function	Runs, catalyzes, freezes
Primitive concepts	Properties are necessary but not sufficient	All cats have tails but not all tails belong to cats
Defined concepts	Properties are both necessary and sufficient	All cats purr and only cats purr If mew→Then a cat
Constraints	Data type, cardinality, domain, range	String, max=1, Biology, 0 <= X <= 10
Instances	A concrete member of a class	Concept = person Instance=Mary Parmelee
Nominal	Concepts not instantiated by type	Italian cat = cat born in Italy
An ontology	concepts+properties+axioms+values+nominals	Ontoweb Portal
knowledgebase	ontology+instances	Clinical domain KB
Knowledge based system	Ontology+instance+inference engine	Cyc Knowledgebase
Metalanguages	Data structuring framework for ontology expression	relational model, frames, description logics

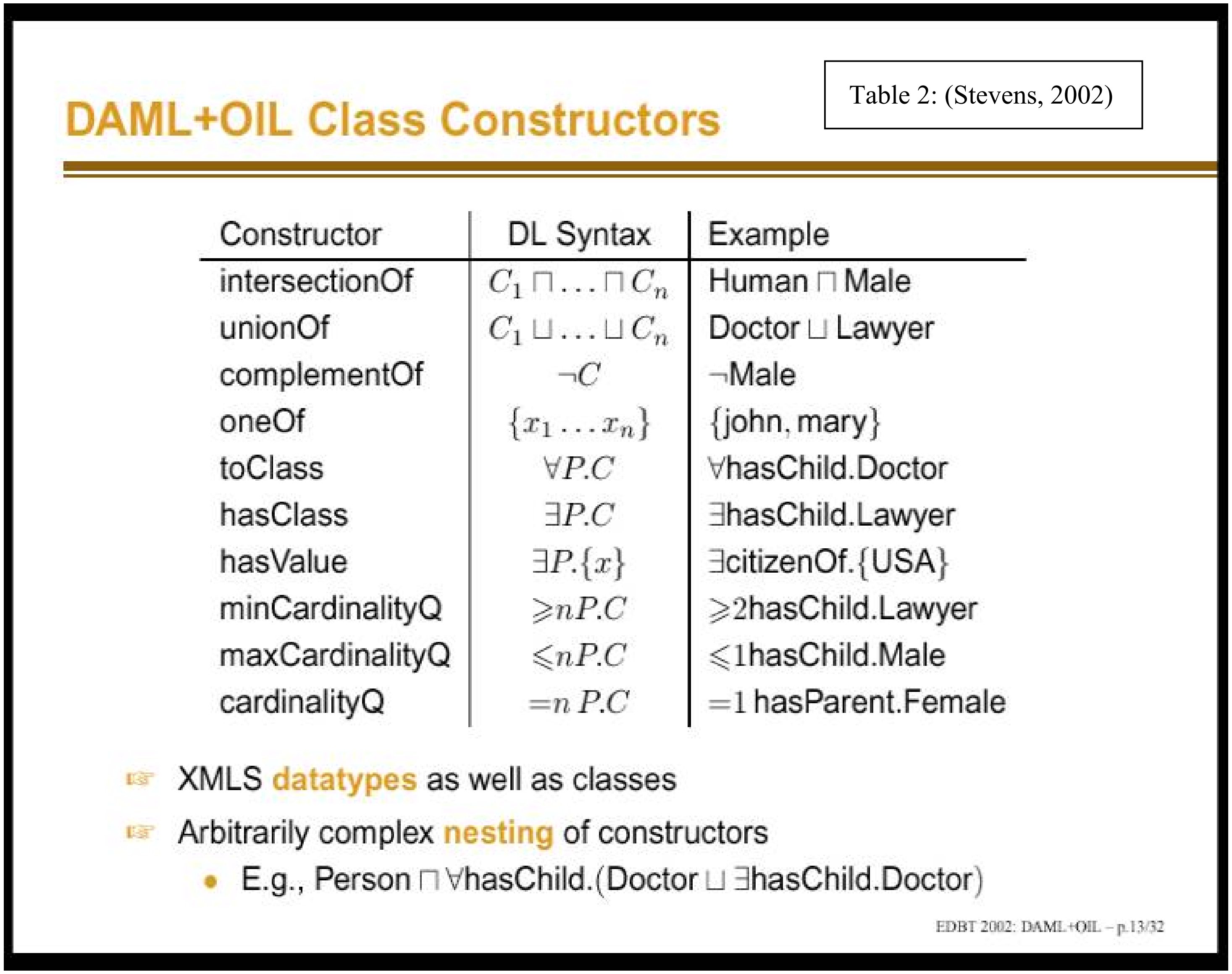


Table 2: (Stevens, 2002)

DAML+OIL Class Constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geqslant n P.C$	$\geqslant$2hasChild.Lawyer
maxCardinalityQ	$\leqslant n P.C$	$\leqslant$1hasChild.Male
cardinalityQ	$= n\ P.C$	=1 hasParent.Female

- XMLS **datatypes** as well as classes
- Arbitrarily complex **nesting** of constructors
  - E.g., Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)

EDBT 2002: DAML+OIL – p.13/32

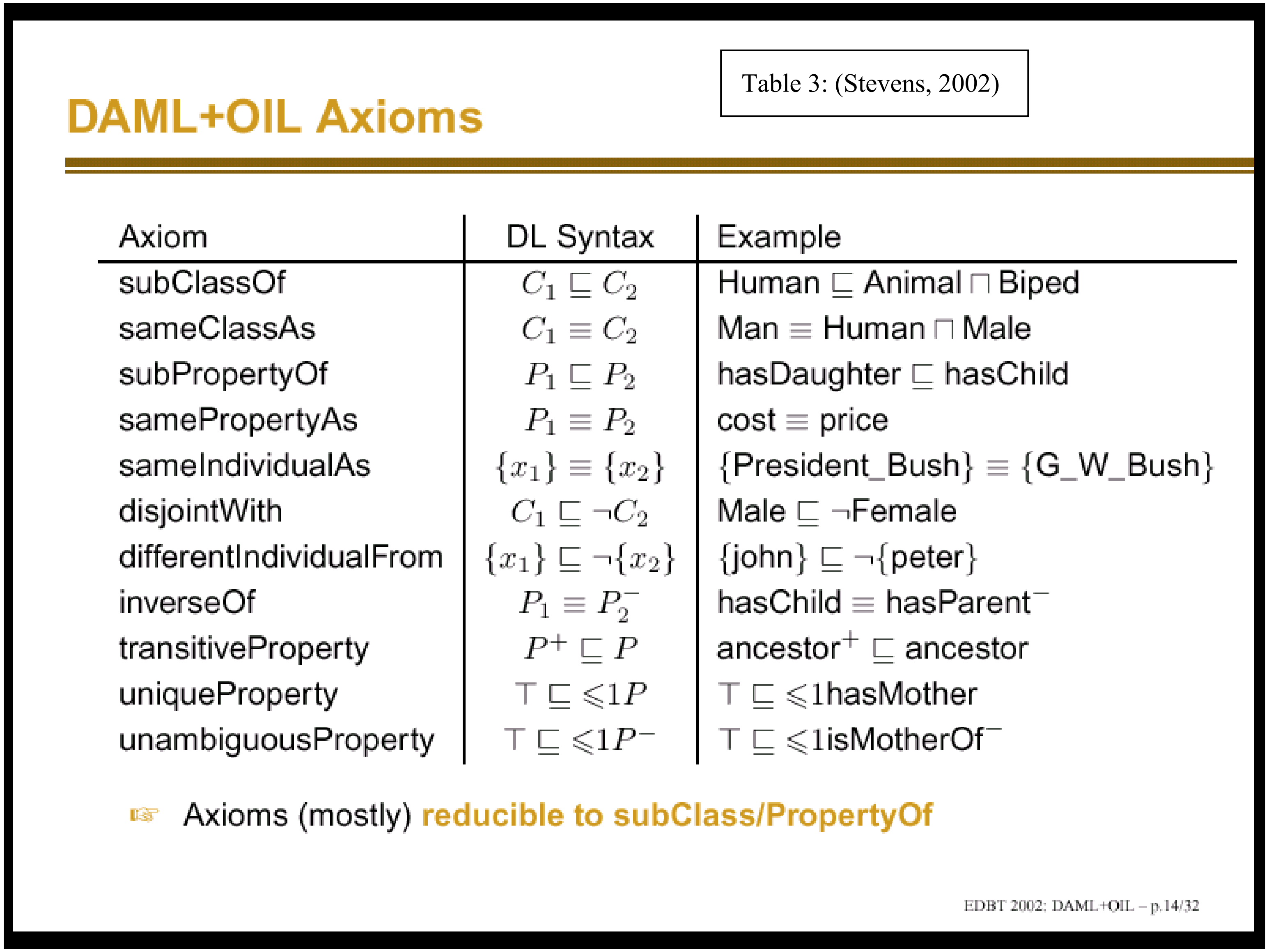


Table 3: (Stevens, 2002)

DAML+OIL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ ¬Female
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ ¬{peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$isMotherOf$^-$

☞ Axioms (mostly) **reducible to subClass/PropertyOf**

EDBT 2002: DAML+OIL – p.14/32

3.2.4 Other metalanguages

Ontologies may also be expressed in various types of markup languages that were not specifically designed to represent ontologies. The most prevalent of these is the Resource Description Framework (RDF), a semantic markup language that was designed to facilitate resource discovery and semantic interoperability of World Wide Web (WWW), resources. RDF is a sanctioned standard of the World Wide Web Consortium (W3C), and is the primary language associated with the W3C Semantic Web project. Because RDF was designed to represent the semantics of existing heterogeneous WWW resources, it is a highly flexible and semantically expressive language.

The cornerstone of RDF is the RDF statement or triple, which captures semantics by linking resources in a subject, (has) predicate, (points to) object format. RDF data models are visualized as graphs that reflect the semantic connections between resources. Figure 6 (Lassila,1999)is an excerpt from the RDF model and syntax specification that illustrates a basic RDF triple statement and it's associated data model that represents the sentence "Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila". In the data model, the sentence subject is represented as an oval, the predicate as the connecting arc between subject and object, and the object as a rectangle. The arc is directional, with the arrow always pointing from the subject to the object of the sentence.

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

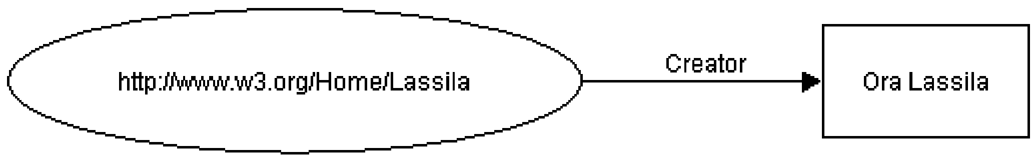


Figure 6: RDF statement and graph model

3.3 Ontological Methodologies

There is no standard methodology for building ontologies. There are a variety of stage-based models, and a few evolving prototype models. Yet there are a number of commonalities that roughly correlate to standard systems analysis practices. Regardless of the particulars of each model the following phases apply to most existing methodologies (Stevens, 2000):

- Setting domain and range: most methodologies develop at least a loose representation of the reference domain boundaries in order to determine the ontology's purpose and scope. These may range from informal verbal definitions to formal specifications including functional requirements.
- Knowledge Acquisition: The process of acquiring the knowledge that will sufficiently represent the reference domain. The main sources are domain

experts, but may also include documentation, artifacts, observations, relevant literature and other ontologies.

- Conceptualization: This is the backbone of the ontology. It is where the bulk of the logical assumptions and inferences are determined which enables the ontology to take semantic shape. It involves identifying key concepts that are representative of the domain, defining them based on their properties determining the relationships that hold between them, and thereby structuring domain knowledge into a conceptual schema.
- Documentation: Creating an informal specification of the ontology including concept definitions, and notation of the logical assumptions that identify the properties and relationships. Ideally this process would integrate standardized definitions and cite their sources as well as provide a detailed documentation of the methodology.
- Formalization: Formalizing the ontology in a machine readable format. This is typically an ontology metalanguage but could also be a database schema and associated programming code.
- Testing: This involves prototyping the ontology for purposes of testing based on a given set of criteria. Criteria might be in the form of competency questions, verifying requirements, or comparing the ontology's performance against other types of ontologies.

3.3.1 Stage based models

Stage based methodologies can be divided into three strategy types based on where the organization process begins in terms of the concepts representing the source or reference domain: top down, bottom up and middle out. In this section, I will describe each strategy in the context of example ontologies to which they apply.

3.3.1.1 Top down strategy

As the name implies, top down strategies are built on a general to specific axis. They begin with broad, high level concepts, then map increasingly more specific concepts and their attributes into a into a type-subtype hierarchy. Maedche and Staab (2001) claim to "capture the commonalities of virtually all ontology models" with a frame based top down strategy of evaluation they call "a semiotics view of ontologies". This strategy defines an ontology as a lexicon containing two reference functions that link terminology, syntax, and semantics: a set of concepts including a single root concept classified in the form of a taxonomy; and template slots, which represent relations specifying a concept-relation or domain-range pair.

Its overall structure is a lexicon containing the union of sets of signs for concepts and relationships. Elements of the formalization process satisfy both reference functions. The taxonomy serves as syntactical reference function linking the terminology to a structured syntax, while the concept-relation pairs provide semantic constraints that give the structure meaning. All concepts are taxonomically related by transitive inheritance of root characteristics, meaning each sub-type concept inherits the characteristics of its supertype or parent concept. Finally, the template slots can be satisfied indirectly by range of value restrictions affecting properties.

3.3.1.2 Bottom up strategy

Bouaud's "Methodological Principles For Structuring Ontologies" suggests that ontologies should be built from the bottom up in a tree hierarchy, leading to the root of "Entity". Bouaud proposes that we must first fix the meanings of terms or "normalize" them on order to compensate for the computer's lack of ability to interpret semantically. This process is called "fixing the necessary intensional conditions of types." In order to normalize terms we "type" them by relating them to domain concepts or categories of thought and then making subsumption-based inferences that rely on the meaning of these category types. A category type is defined intensionally in terms of necessary and sufficient conditions, then it is expressed extensionally in its essential characteristics or properties. These types become the basic modules or building blocks of the ontology.

Following normalization, building of the ontological tree structure is guided by four principles of structural organization. First, the similarity principle defines the basic meaning of a taxonomic link. It states that a "child" type inherits the characteristics of the type that subsumes it, called its "parent" type. The specificity principle states that a child type must have a "specific difference" from its parent type, a distinctive property that the parent type lacks. Third, the opposition principle, organizes siblings in "opposition types" that are both distinct and incompatible with each other. The final principle, the unique semantic axis principle, combines principals one and two in requiring that all (child) subtypes differ from their (parent) supertype by a common property or axis, and from sibling subtypes by a unique property or value (Bouaud,1995).

The resulting tree structure is a subsumption hierarchy in which the semantic meaning of each type is dependent upon both its position in the tree and its intensional distinction from other types (Bouaud, 1995). In this manner, the semantic infrastructure is built into the controlled vocabulary itself, and defined by the total set of relations between classes and properties. Moreover, deriving the ontological terminology from existing vocabulary and then building the semantic rules of inference into this vocabulary, maximizes the communication function, while minimizing ontological restrictions (Jacob, 1997).

3.3.1.3 Middle out strategy

The middle out life cycle is modeled by Uschold and King's "Enterprise Ontology" (1995, as cited in Jones, et al, 1998). In the first stage the ontologist identifies a clear scope and focus for the ontology via a written statement of purpose. This is followed by a "middle out" scoping process that begins with identifying the core (key) concepts in an ordered list. Then the hierarchy is built out from these concepts through generalization or specialization. Next, the ontology is formalized or coded into a formal language. Finally, the ontology is evaluated based on one or more frames of reference, such as practical application in the form of implementation testing.

The first step of this approach contrasts with evolving prototype models such as the Methontology and IDEF5 models, which have specifications, but no clear statement of purpose (Jones et al, 1998). One weakness of the "Enterprise Ontology's" strategy is its lack of guidelines for identifying key concepts (Jones et al, 1998). However, this process is easily supplemented with guidelines adopted from current ontological analysis such as Bouaud's "Methodological Principles for Structuring an Ontology" (1995), and Guarino's notion of sortal vs. non-sortal predicates (as cited in Jones, et al, 1998).

The "Toronto Virtual Enterprise" (TOVE) ontology, developed by Grunninger and Fox (1995, as cited in Jones, et al, 1998) shares a stage based, middle out strategy with the "Enterprise Ontology". However, TOVE's fundamental distinction from other ontologies lies in it's definitional processes. TOVE identifies ontological concepts and their relational terms through indirect specification in the form of "competency questions", while the "Enterprise Ontology" use a more direct, brainstorming approach (Jones, et al, 1998). The former bases ontological terminology on the language of the developing ontologist, while the latter defines ontological terminology based on the existing language of the domain. Another distinguishing feature of TOVE is that it defines terminological semantics directly with stated axioms, while the "Enterprise Ontology" implies semantic rules of inference via the coherence of the relational structure rather than stating them directly (Jacob, 1997). Lastly, because TOVE bases the ontology on the task of question formulation, its potential for reusability is limited (Jones et al, 1998).

Lastly, it's important to note that ontologies are rarely fully defined in a single linear process. Inefficiencies in the form of inaccuracies or omissions will likely emerge with further implementation testing. These must be corrected through extension or refinement of the ontology. This "unspecified feedback loop" highlights the fact that although stage-based approaches appear to be linear, building ontologies is always an iterative and dynamic process.

3.3.2 Evolving prototype strategy

The quintessential evolving prototype methodology is the Methontology, which advocates simultaneous knowledge acquisition, integration, documentation, and evaluation of the ontology. What makes this methodology unique is that it is designed as a domain expert ontology development guide. The conceptualization is prototyped by creating a set of intermediate representations that "bridge the gap between how people think about a domain and the languages in which ontologies are formalized" (Blaquez et. al., 1998).

The methodology includes detailed instructions on building these intermediate representations of the ontology. Since the modeling process is incremental, it builds from simple associations to more complex iterations. This methodology, enables domain experts who are unfamiliar with ontology development methods to build ontologies themselves (Blaquez et. al., 1998).

An ontology editor further guides the domain expert in the development process and automatically codes the ontology in a knowledge representation language. Automatic encoding both expedites the formalization process, and enables the domain expert to transform the ontology into machine readable form without learning complicated coding languages. Figure 7 is an overview of the Methontology process.

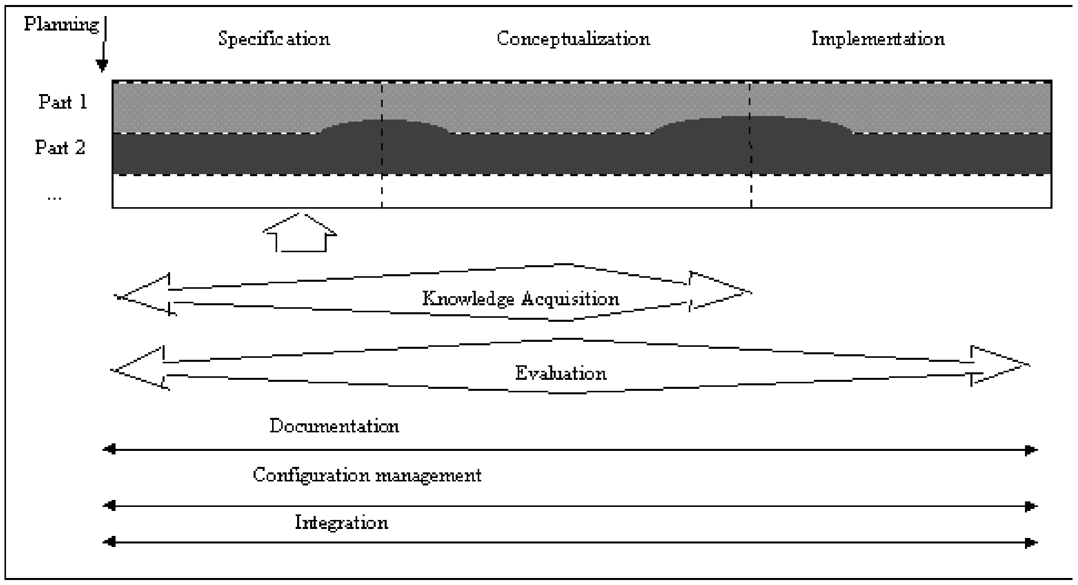


Figure 7: Methontology development process

3.4 Ontological Perspectives

The majority of ontological methodologies conceptualize based on an object oriented world view of the reference domain, centralizing the ontology around concepts that represent the things or objects in a domain. Recently however, a new emphasis on modeling more varied world views have extended the functionality and application of ontologies to include process, meta-ontologies, and semantically integrated ontology networks.

3.4.1 Process ontologies

Process ontologies take a process rather than object-oriented world view of the reference domain. Although process ontologies are not new, they are gaining more attention recently as appropriate methodologies for optimizing the semantic expressiveness of process oriented domains. Process ontologies essentially invert the object oriented world view. It essentially groups objects based on the common processes that they apply to rather than their common definitions of existence. This enables organization of a reference domain based on the meaning of its function or what it "does", rather than what it "is". Figure 8 diagrams a process ontology that was developed by developed Klein and Bernstein (2002) have a process ontology methodology that models web services for the purpose of facilitating information retrieval. Process ontologies can be derived from existing process models such as work flow models using standard modeling primitives such as tasks subtasks, resources, inputs, outputs, and exceptions.

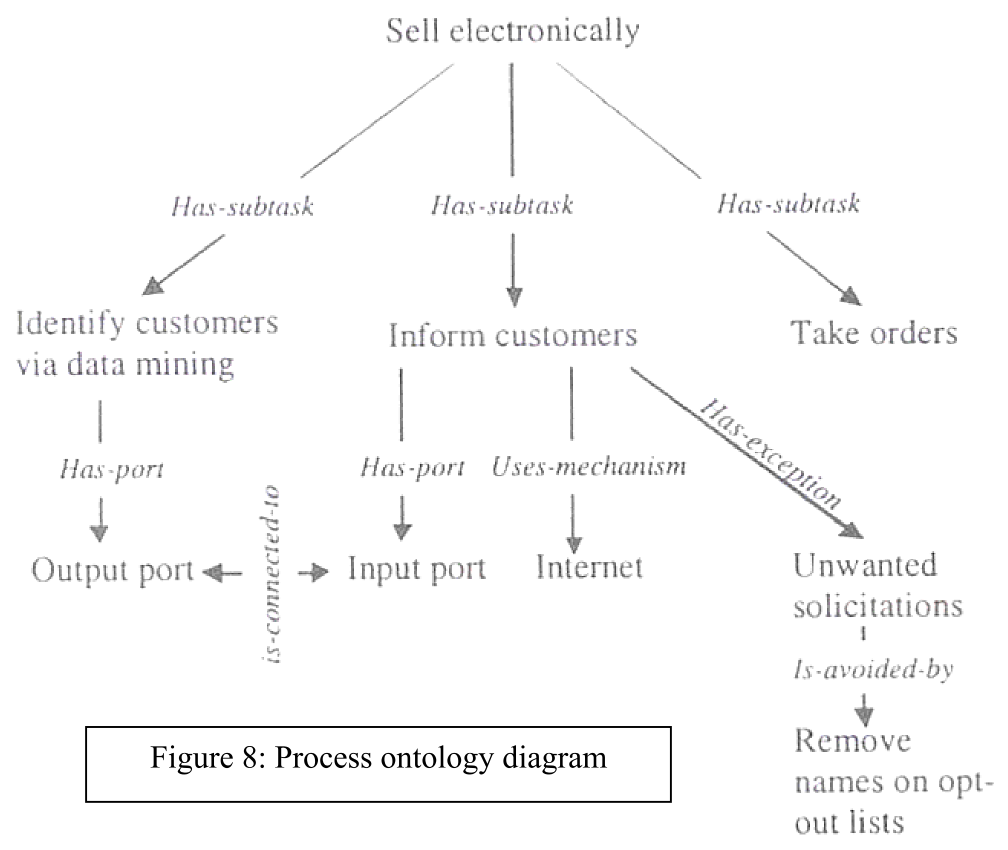


Figure 8: Process ontology diagram

The key components of a process based ontology as defined by Klein and Bernstein (2002) are:

- Attributes: Processes can be annotated with attributes that capture such information as a textual description, and typical performance values such as how long a process takes to execute.
- Decomposition: A process can be modeled as a collection of processes that
- can in turn be broken down or “decomposed” into sub-processes.
- Resource Flows: All process steps can have input and output ports through which resources flow. One innovation we use is to recognize that

processes can be divided into ‘core’ activities as well as those involved in coordinating the flow of resources between core activities This insight allows us to abstract away details about how sub-processes coordinate with each other, allowing more compact service descriptions without sacrificing significant content.

- Mechanisms: Processes can be annotated with the resources they use (as opposed to consume or produce). For example, the Internet can serve as a mechanism for a process.
- Exceptions: Processes typically have characteristic ways they can fail and, in at least some cases, associated schemes for anticipating and avoiding or detecting and resolving them. This is captured by annotating processes with their characteristic ‘exceptions’, and mapping them to processes describing how these exceptions can be handled.

Another factor that Klein and Bernstein stress is the notion of accounting for every possible use for a given process. They address this issue with a specialized process query language (PQL) that defines query algorithms to retrieve processes not explicitly indexed as serving a given purpose. They accomplish this using query mutation operators (2002).

3.4.2 Upper level ontologies

Upper level ontologies are important for ontology sharing and integration of domain ontologies. They map high level generic or core concepts that represent interdomain commonalities. They can include generalizations of process or task as well objects and concepts. Figure 9 shows the Sowa's upper ontology (Sowa, 2001)

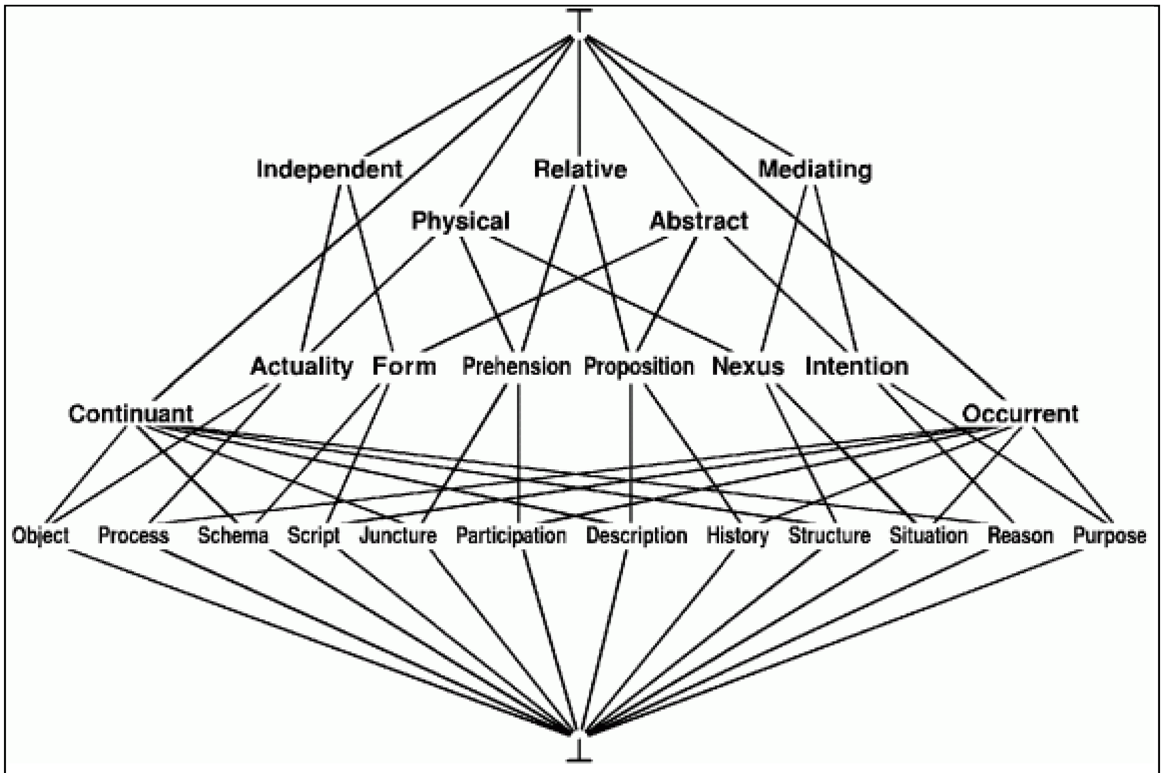


Figure 9: Sowa's Upper Level Ontology

3.4.3 Meta-ontologies

Meta-ontologies are ontologies that conceptualize other ontologies. They can be object or process oriented. Meta-ontologies describe metadata about ontologies and their associated elements. Meta-ontologies can also be used as a sort of ontology prototyping system for capturing changes of an evolving ontology called an evolution ontology. Figure 10 is an example of a meta-ontology that models interoperability issues between two ontologies called the Semantic Translation Ontology or RDFT (Stevens, 2002).

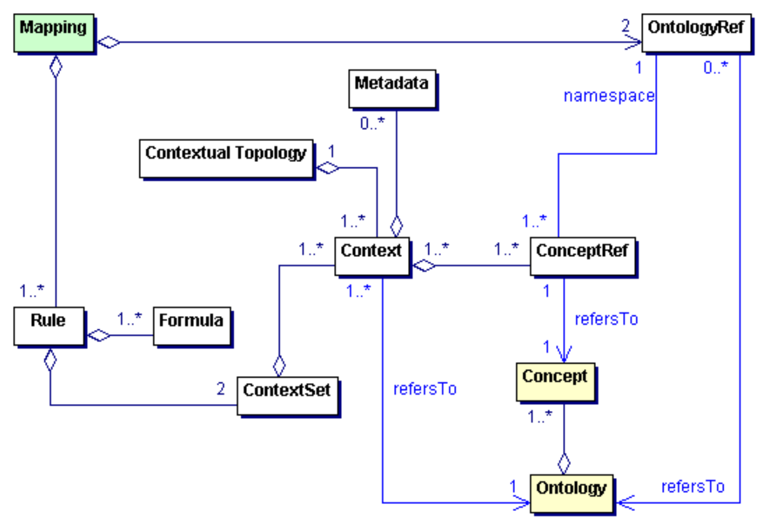


Figure 10: Meta-ontology example

3.4.4 Integrated ontologies

Integrated ontologies are concerned with combining and blending existing domain ontologies that were not originally designed for interoperability. The source ontologies may have conflicting formats, and may be defined using incompatible approaches or different domains. The goal of integrative ontologies is to transform disparate source ontologies into a common view of the application domain (Missikoff, 2002). The European Information Society Technologies (IST) project has developed the "Harmonize" Local As View (LAV) approach to ontology integration.

Harmonize researchers have identified two main groups of "clashes", that they attribute to "differences in the conceptual schemas of two applications attempting to cooperate"(Missikoff, 2002):

- Lossless clashes: are clashes that can be reconciled without loss of information. Examples include:
    - naming clashes, where two terms represent the same concept
    - structural clashes, where information elements are grouped differently
    - unit clashes, where the same concepts are represented with different units of measure.

- Lossy clashes: cannot be reconciled without loss of information. These usually involve Precision gaps, where the same concept is represented at different levels of granularity. Examples include:
    - Measured vs. approximated information.
    - Specific versus general descriptors such as 'structural protein' vs, just 'protein'.

Harmonize has developed a frame-based ontology integration metalanguage called the Object Process Actor modeling language (OPAL) and associated ontology management system called SymOntoX. The SymOntoX interface shows an integrated domain view that divides ontologies into layers representing levels of concept granularity. The top layer or upper domain ontology (UDO) contains high level concepts such as 'actor', 'process', 'event' and 'goal'. The bottom layer or lower domain ontology contains primitive concepts such as Dalton or molecule.

It is relatively easy to find common or agreed upon definitions at the upper and lower end of the spectrum. The more divergent middle layer contains the application ontology. This ontology can vary dramatically between applications depending on the problems encountered, logic used, methodology, cultural differences and underlying technology, which "often contaminates the conceptual model" (Missikoff, 2002). For instance, nomenclature problems are likely to surface here for a genomic application.

Harmonize comprises and provides for the domain expert, a descriptive framework called an Integrated Meta Ontology (IM) for a given domain. This framework consists of set of upper and lower level concepts derived from domain industry standards. Participants who would like to exchange information without changing their proprietary schema can simply annotation their local conceptual schema (LCS) with the concepts provided by harmonize. Annotation involves assembling primitive constructs or concepts from the IM to map defined concepts in the LCS.

The annotated LCS provides a semantic reference for each local schema for the purpose of information exchange. In this manner the harmonize system acts as an ontology translator enabling participants to both export and import information according to their unique schema. Local schemas are transformed to a common representation for exchange called the Harmonise Interoperability Representation (HIR). Harmonize is currently developing a model for the European tourism industry (Missikoff, 2002). Figure 11 is an overview diagram of the harmonize approach to ontology integration.

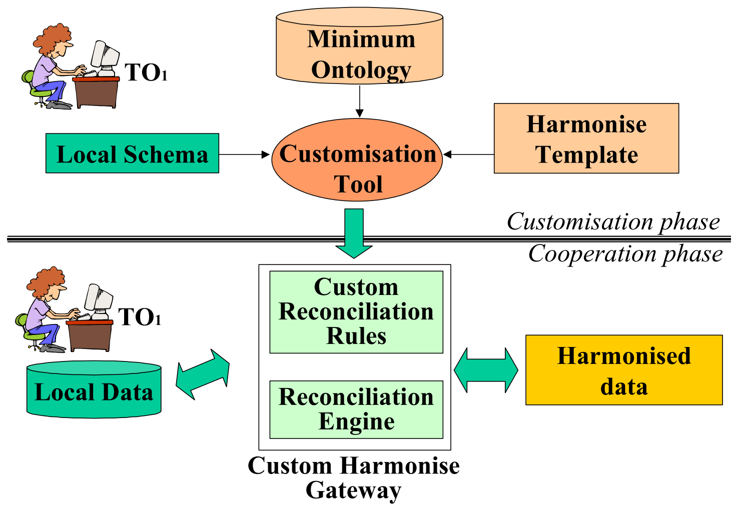


Figure 11: Harmonize ontology integration system

4 DATABASE METHODOLOGIES

Database methodologies are essentially logical collections of constructs used to define and represent data structure and relationships. They include conceptual models that represent logical data representation, and physical models that represent how the data is structured in the database (Yang, 2002). Although biological databases encompass a wide range of database methodologies, in general, most major projects use either the relational or object-oriented approach. In this section, I will provide a brief overview of relational, object oriented and hybrid databases and their methodologies

4.1 Relational Databases

Relational databases are represented as a collection of data entities, stored in tabular format. Each table is called a relation and is stored as a separate file in the database. Relations hold sets of entities that are semantically linked by their common characteristics. Relations are linked to each other by their unique identifiers or primary key attributes.

4.1.1 Entities as tables

An entity is a person, place, thing or event about which data is being stored.
An attribute is a characteristic or property of an entity. Each row or tuple in a relation represents an entity and each column an attribute of that entity. The semantic linkage within a relation is represented by the intersecting field between an entity and its characteristic or attribute (Yang , 2002). The intersection value represents a single characteristic of the associated entity. Relations apply no

syntactic constraints to attribute order. Figure 12 illustrates the structural representation of a relation (Berners-Lee, 2002).

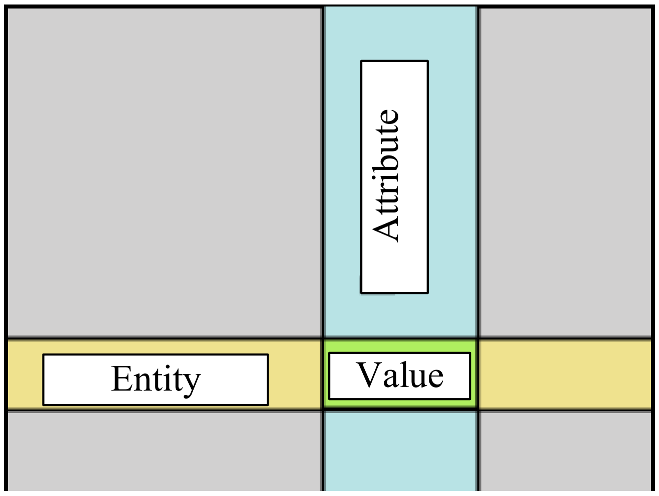


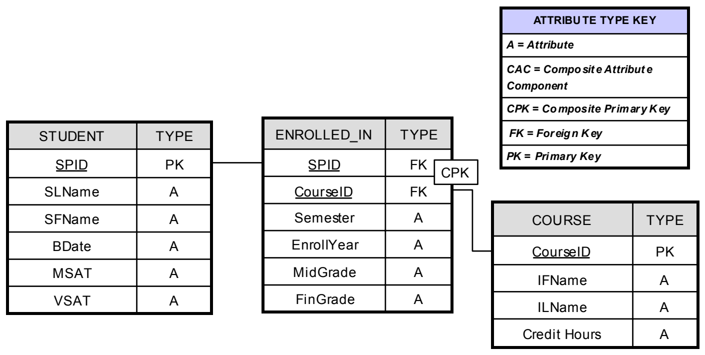
Figure 12: Relation example

The tabular structure of the relational database constrains semantic linkages to binary relations of primitive or atomic entities. Other intrarelation constraints include (Yang, 2002):

- Attributes have range of values called a domain
- Attributes within a column must have the same data type (eg. string, integer)
- Entities (rows) must be uniquely identified by one or more key attributes called a primary key (entity integrity)
- Key attributes must not depend on other attributes to describe the entity
- Non-key attributes are functionally dependent on key attributes

4.1.2 Interrelational structure

Relational models also represent interrelations or relationships between tables. Semantic expressiveness is supported in relational models primarily by linking related tables by their key attributes to form an interconnected network of relations. While uniquely identifying entities with primary keys ensures entity integrity, linking tables with primary keys, called foreign keys, ensures referential integrity. This means that tables having foreign key dependencies or matching foreign keys in related tables, can not be deleted, hence the semantic linkages are preserved. Figure 13 represents a simple relational model with primary to foreign key relationships that depicts a student, enrollment, course relationship (Yang , 2002).



ATTRIBUTE TYPE KEY
A = Attribute
CAC = Composite Attribute Component
CPK = Composite Primary Key
FK = Foreign Key
PK = Primary Key

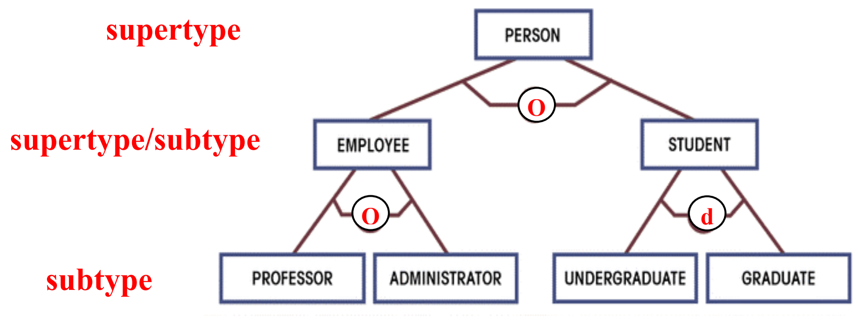
STUDENT	TYPE
SPID	PK
SLName	A
SFName	A
BDate	A
MSAT	A
VSAT	A

ENROLLED_IN	TYPE
SPID	FK
CourseID	FK
Semester	A
EnrollYear	A
MidGrade	A
FinGrade	A

COURSE	TYPE
CourseID	PK
IFName	A
ILName	A
Credit Hours	A

Figure 13: Relational model

Basic hierarchical constraints such as supertype-subtype relationships, as well as mutually exclusive (disjoint) and overlapping sibling relationships are also supported. Figure 14 diagrams an example of hierarchical constraints describing faculty and student types of people (Yang , 2002).



O = overlapping
D = disjoint

Figure 14: Relational hierarchy

4.1.3 Query languages

The structure of the relational model provides a framework of primitive semantic linkages from which more complex semantic relationships are expressed through query languages or programming. Based on relational algebra rules, queries filter large sets of relational data by selecting specific rows and columns, applying parameter constraints that return data within a specific range of values, and deriving data via functions. A sample query for the relational model in Figure 14 might be: "What are the names of all students who are enrolled in a theology course with course number T2?"

This query would take three steps:

- Find CourseID =T2 in the COURSE table

- Find CourseID =T2 in the ENROLLED_IN table and return all associated student (SPID) numbers
- Find student (SPID) numbers in the STUDENT table and return associated Student Names

4.2 Object Oriented (OO) Databases

Object oriented database design combines database design with object programming language. OO defines a database as a set of objects, properties and operations.

4.2.1 Entities as objects

Database Objects are modeled as OO programming language objects. While relational databases fragment complex data into pieces of atomic or primitive data, by storing data together with the methods to access that data, called encapsulation, object oriented databases are able to store complex data types. They also support a wide range of data types including user defined data types.

4.2.2 Relationships and navigation

Object oriented design supports four types of semantic relationships: inheritance or type-subtype, association, whole-part (aggregation), part –whole (inverse) (Chandler and Hand, 2002).

Querying of objects, called navigation**,** involves searching objects for pointers to other objects. This paradigm is more semantically expressive and flexible than the relational model. Figure 16 illustrates the same student, enrollment, course relationship as the relational model in Figure 15. This example shows two student objects and a course object, the enroll relation is handled with code and interobject pointers. The student objects have information about what courses that student is taking and the course object has information about which students are enrolled in that course (Chandler and Hand, 2002).

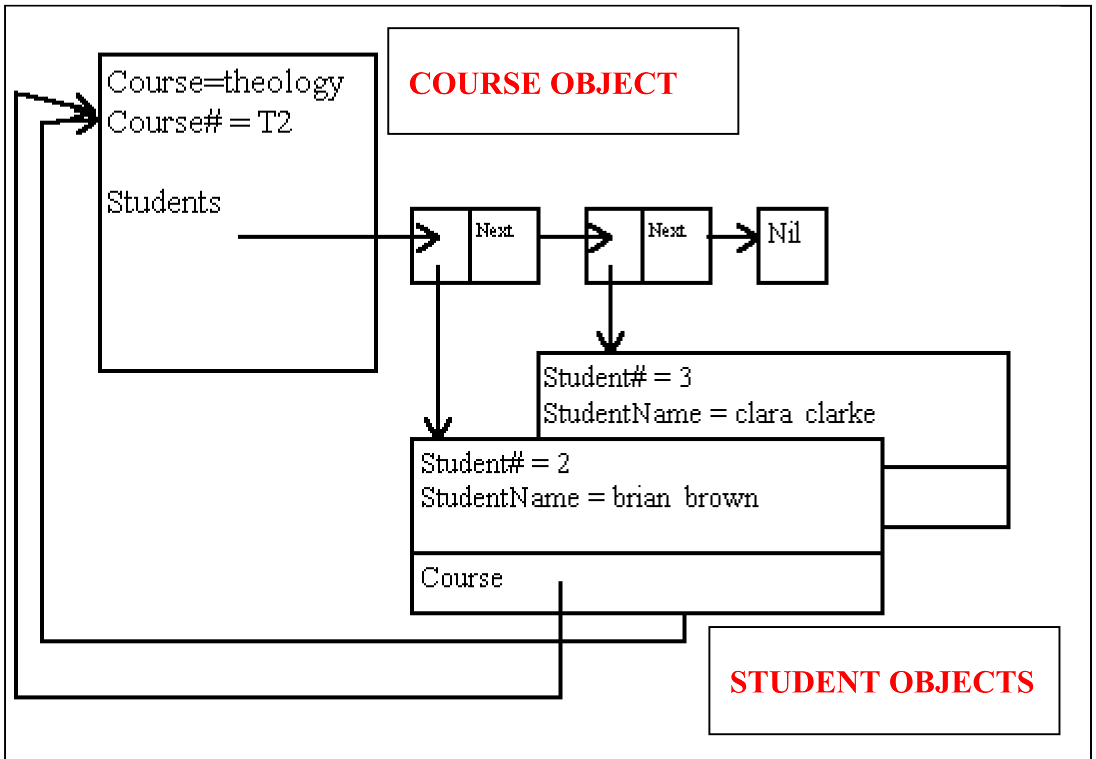


Figure 15: Object model example

Querying for the names of all students enrolled in course number T2 would be a two step process written in the same programming code as the defined objects:

- Search the COURSE Index and find CourseID=T2)

- Follow Student Pointers to STUDENT object and return each associated student name

4.3 Relational to Object Oriented Comparison

Since the basic relationships between entities are built into the data structure of relational databases, in general they are best applied to domains with a large number of data instances, but simple relationships. Object oriented designs are more suited to complex data schemas because interobject relationships are handled totally with navigational algorithms (Fernstrom, 2000). Table 4 is a relational object oriented comparison that was written by Christer Fernstrom as a technology briefing report for the RENAISSANCE ESPIRIT Project, a database research project at Lancaster University (2000).

Table 4: Relational to Object Oriented Database Comparison

	RELATIONAL	**OBJECT ORIENTED**
complex data types	- supports binary large objects (BLOBs) which are essentially very large database columns - database columns must generally be simple data elements, rather than structures and arrays	- supports complex data types, including user defined data types
locating data	- fast at locating simple data types - no support for searching or indexing BLOBs	- usually slower than RDBMS when locating simple data types - sophisticated algorithms used to search for data within complex data types
language standards	- Major RDBMS's use a standard language (SQL2 / SQL92) with vendor-specific extensions. SQL is both a data definition language (DDL) and a data manipulation language (DML).	- Many ODBMS's support object query language (OQL) and object definition language (ODL) which are part of the larger Object Data Management Group (ODMG) standard. There are varying degrees of compliance amongst ODBMS vendors with the overall ODMG standard. - unlike RDBMS's which use SQL Insert, Update, Delete, data manipulation is performed in the programming language by invoking methods on the object -- this allows for better object encapsulation.
functions / methods on DBMS	- provides triggers / stored procedures which are written in the native DBMS language - function / method inheritance not supported	- object methods are typically written in a selected OO programming language (usually C++, Java or Smalltalk) - object/method inheritance is supported

4.4 Hybrid Databases

In the past five years, most major relational database management systems (RDBMS) have incorporated basic object features such as triggers and binary large objects (or BLOBs). These features are designed to handle the increased application complexity required to manage complex data types such as audio and video as well as the demand for more sophisticated searching indexing and query optimization for complex data types. The result is a hybrid system called an Object Relational Database Management System (ORDBMS) or universal servers. Some of the key features that ORDBMSs provide are user defined data types and functions, enhanced search and indexing features, and expanded support for programming languages (Fernstrom, 2000)

5 ONTOLOGY-DRIVEN DATABASE APPLICATIONS IN BIOLOGY

There are a numerous ontology driven biological database projects currently under development worldwide. For example, model organism databases (MODs) such as EcoCyc model specific organisms, controlled vocabulary (nomenclature) projects such as the Gene Ontology (GO) are concerned with semantic interoperability among the biological community, and pathway databases model metabolic pathways of specific organisms. While describing the range of development methodologies of these projects is beyond the scope of this study, I will summarize two representative projects that characterize leading methodological approaches: the GO nomenclature database and the EcoCyc Pathways knowledge based system.

5.1 The Gene Ontology

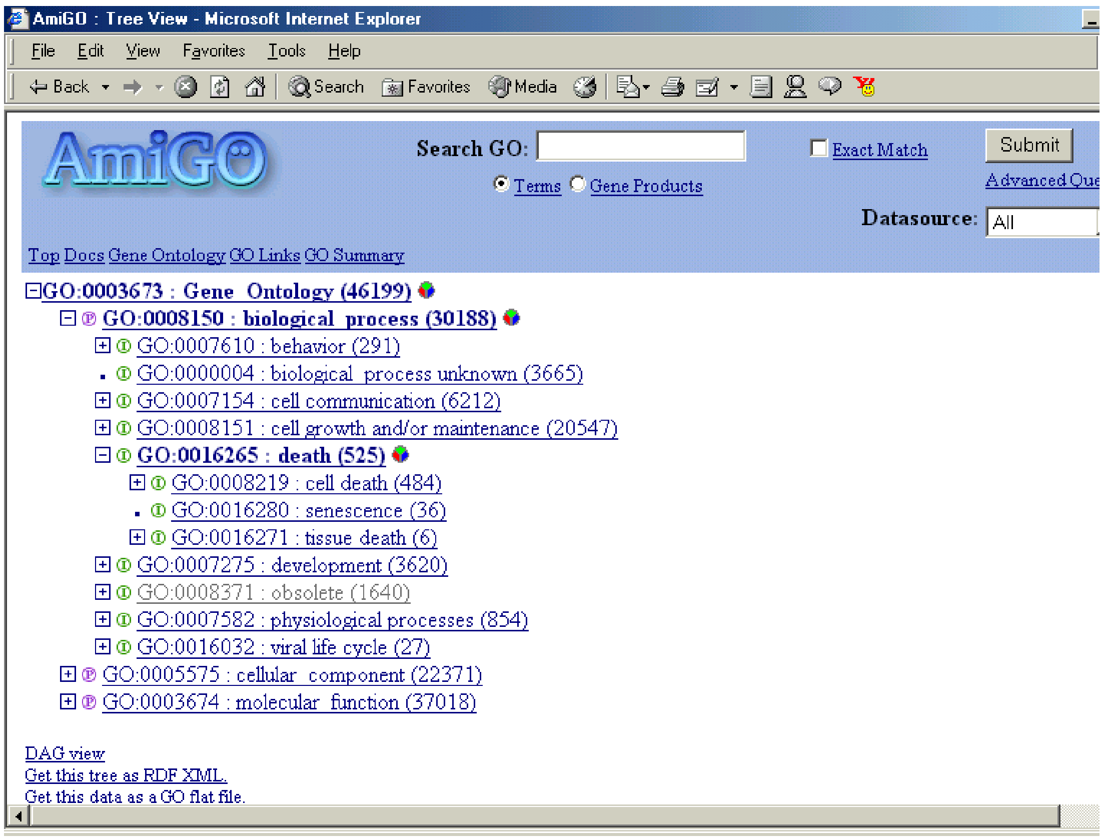
GO was developed by a consortium of MOD database curators for the purpose of resolving gene nomenclature issues in gene related database annotation. Instead of attempting to reconcile the existing nomenclature system, GO aims to develop a consistent annotation system that describes and relates genes based on their characteristic functions rather than their organism of origin.

The goal of the GO project is to address the gene nomenclature problem by developing a standardized controlled vocabulary to describe the function, processes, and location of a gene product for use as standardize gene annotation across model organism databases. Curators from database projects such as the *Drosophila* community's FlyBase, the *Saccharomyces* Genome Database, and the Mouse Genome Database, joined forces to develop a centralized database of terms that models the function of genes in all organisms (Pearson, 2001).

5.1.1 Ontological methodology

GO uses a frame-based process ontology methodology as the basis for database schema development. GO is actually three separate ontologies that specify gene function, process, and cellular location, with a combined total of over 10,000 concepts. These upper level concepts are organized into type-subtype and class-property hierarchies using the frame defined relationships `is a kind of" and `is part of ' to specify gene product roles (Stevens, 2002).

Every concept in the GO ontology is manually assigned by curators who attach gene instances to GO terms and resolve any nomenclature collisions. Common issues such as homographs referring to different processes are typically resolved by adding disjoint subtypes to the representing hierarchy. GO coordinator Midori Harris describes the process, "The vocabularies are dynamic - a work in progress" (Stevens, 2000). Figure 16 is an example of the GO as rendered in the AmiGO ontology browser (GO, 2002).


AmiGO : Tree View - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
AmiGO
Search GO:
Exact Match
Submit
Terms Gene Products
Advanced Que
Datasource: All
Top Docs Gene Ontology GO Links GO Summary

GO:0003673 : Gene_Ontology (46199)
- GO:0008150 : biological_process (30188)
  - GO:0007610 : behavior (291)
  - GO:0000004 : biological_process unknown (3665)
  - GO:0007154 : cell communication (6212)
  - GO:0008151 : cell growth and/or maintenance (20547)
  - GO:0016265 : death (525)
    - GO:0008219 : cell death (484)
    - GO:0016280 : senescence (36)
    - GO:0016271 : tissue death (6)
  - GO:0007275 : development (3620)
  - GO:0008371 : obsolete (1640)
  - GO:0007582 : physiological processes (854)
  - GO:0016032 : viral life cycle (27)
- GO:0005575 : cellular_component (22371)
- GO:0003674 : molecular_function (37018)

DAG view
Get this tree as RDF XML.
Get this data as a GO flat file.

Figure 16: GO ontology hierarchy

Although GO supports multiple inheritance type-subtype (multiple types or parents for one subtype) and synonymous relationships, many of the relationships in GO are expressed implicitly or inferred by position in the hierarchy (Stevens, 2000). This approach imposes syntactic dependencies that will likely result in significant scalability issues for direct database schema implementation in rapidly changing domains. For relational schemas direct mapping could easily involve a network of thousands of tables representing specific nodes in the hierarchy, and for object oriented schemas it would involve the programmatic creation of thousands of complex objects and their respective interconnecting persistent pointers.

5.1.2 Transform methodology

GO does not prescribe a specific methodology for ontology to database transform, nor does it recommend implementation in a specific database technology. However, the project has implemented the GO in a My SQL relational database that addresses schema complexity by incorporating an RDF style graph model. This model allows for schema simplification by grouping type-subtype concepts into common tables and denormalizing recursive relationships with a graph path layer that represents all hierarchical relationships in the schema. This is accomplished with two types of graph path tables, one represents the graph nodes that identify two concepts as ancestor and child and assigns them a unique graph ID and the other is represents the graph arcs that assign each term to a graph ID. Figure 17 is an excerpt of the GO schema that illustrates how graph path represents hierarchical relationships.

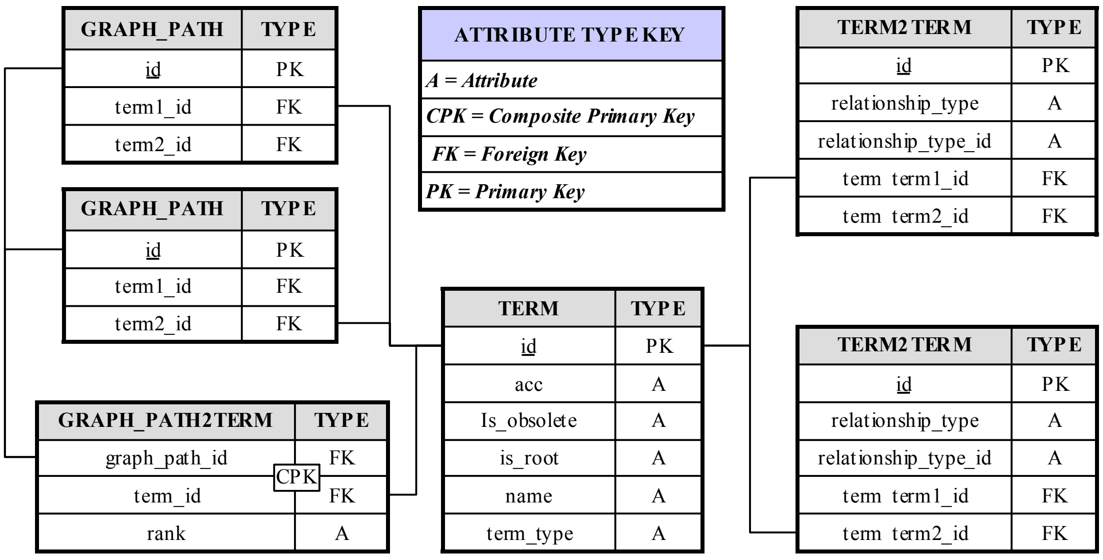


Figure17: GO to graph path relations

This method puts some of the ontology modeling complexity into the data, which greatly simplifies the data model. However, due to the atomic nature of relational database design, one side effect of this method is a more complicated query process in order to reassemble these relationships. For instance, the GO database site provides the following query as an example of how to retrieve all subtypes or child terms of the term 'transmembrane receptor':

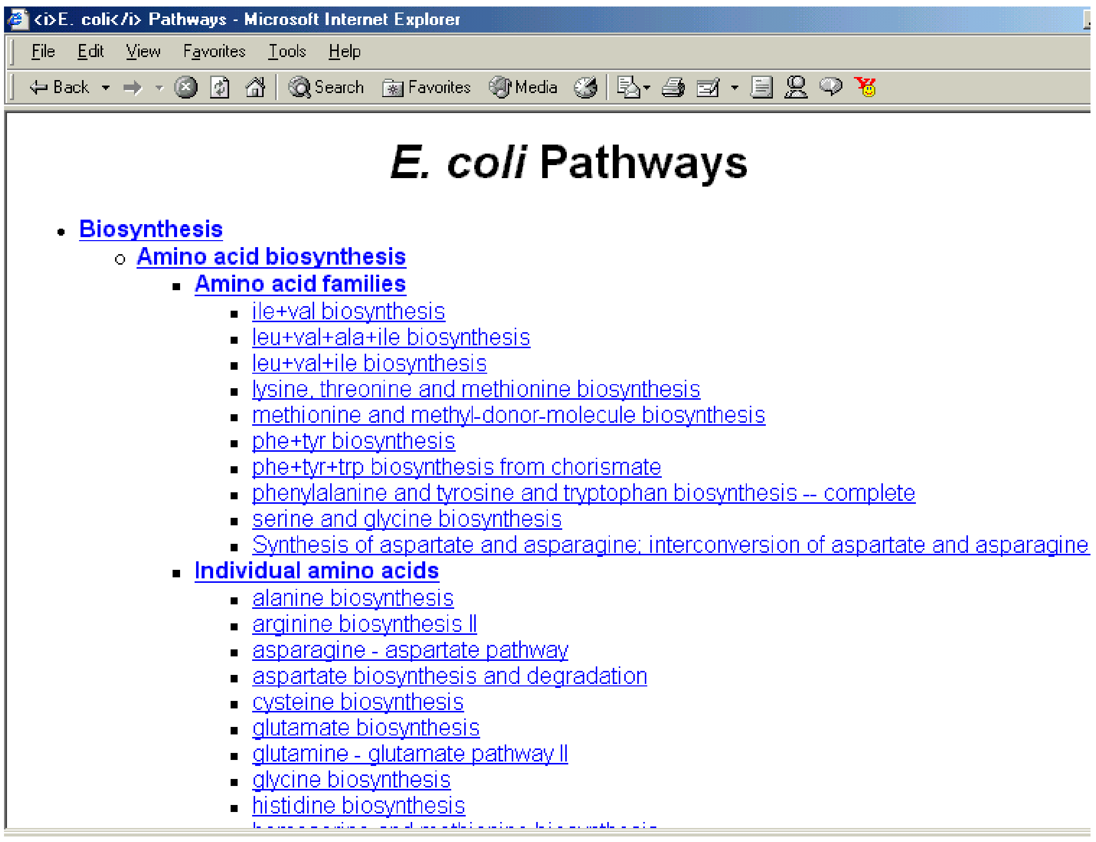
```
SELECT  rchild.*
FROM term AS rchild, term AS ancestor, graph_path
WHERE graph_path.term2_id = rchild.id and
          graph_path.term1_id = ancestor.id and
          ancestor.name = 'transmembrane receptor'
```

5.2 EcoCyc Pathways Expert System

EcoCyc is a bioinformatics MOD that was developed by the Stanford Research Institute (SRI) an non-profit research center. EcoCyc describes the structure and function of the E. coli (*Escherichia coli)* bacterium. The goal of the project is to facilitate a system-level understanding of E. coli.

5.2.1 Ontological methodology

Like GO, EcoCyc uses a frame based ontology. However, EcoCyc uses its ontology as a direct roadmap for an object oriented database schema. EcoCyc version 5.6 has over 1700 concepts that are encoded as objects in the EcoCyc database, including key concepts in biochemistry and molecular biology, their associated properties and relationships among those objects (Stevens, 2000). Figure 18 shows an excerpt of the EcoCyc Pathways object hierarchy.


<i>E. coli</i> Pathways - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media

E. coli Pathways

- Biosynthesis
  - Amino acid biosynthesis
    - Amino acid families
      - ile+val biosynthesis
      - leu+val+ala+ile biosynthesis
      - leu+val+ile biosynthesis
      - lysine, threonine and methionine biosynthesis
      - methionine and methyl-donor-molecule biosynthesis
      - phe+tyr biosynthesis
      - phe+tyr+trp biosynthesis from chorismate
      - phenylalanine and tyrosine and tryptophan biosynthesis -- complete
      - serine and glycine biosynthesis
      - Synthesis of aspartate and asparagine; interconversion of aspartate and asparagine
    - Individual amino acids
      - alanine biosynthesis
      - arginine biosynthesis II
      - asparagine - aspartate pathway
      - aspartate biosynthesis and degradation
      - cysteine biosynthesis
      - glutamate biosynthesis
      - glutamine - glutamate pathway II
      - glycine biosynthesis
      - histidine biosynthesis

Figure 18: Pathways hierarchy example

5.2.2 Transform methodology

5.2.2.1 The knowledgebase

One of the key developers of EcoCyc, Peter Karp (2001) describes the EcoCyc database as follows:

> An interconnected web of frames (objects) stored in a frame knowledge representation system (similar to an object-oriented DB). Each frame represents a distinct biological object (such as a gene or a protein), and the labeled connections between those frames represent distinct semantic relationships among the objects, such as the relationship of a gene to its protein product, or the relationship of a protein to a reaction that it catalyzes.

Karp argues that by encoding scientific theories in symbolic form, we "open new realms of analysis and understanding for theories that would otherwise be too large and complex for scientists to reason with effectively". Based on this analogy, Karp has dubbed EcoCyc a ' computational symbolic theory" which he defines as a database that "structures a scientific theory within a formal ontology so that it is available for computational analysis"(2001). Table 5 describes the number and types of objects in the EcoCyc version 5.6 knowledgebase.

Table 5: EcoCyc Knowledgebase Objects

OBJECT CLASS	OBJECT COUNT
Pathways	165
Reactions	2604
Enzymes	905
Transporters	162
Genes	4393
Transcription Units	629
Promoters	740
DNA-Binding Transcriptional Regulators	100
DNA Binding Sites	854
Citations	3508

Encoding scientific theories in a symbolic knowledgebase refers to the ability of a knowledge based system to infer relationships from the interpretation and synthesis of experimental results. Providing this type of value added summary data can aid scientists in checking theories for accuracy and consistency (Karp, 2001).

5.2.2.2 The production system

Logical reasoning within a knowledge based system is accomplished using expert systems technology usually in the form of a production system. A production system consists of a set of rules and corresponding propositions that the production system recognizes as being true. The rules are in first order logic "IF→Then" form of A^B→C^D. This translates to the system as, If *A* and *B* are present, then *C* and *D.* An inference engine searches for rules that contain the variables on the "If" side, then fires them by adding the variables on the "Then" side.

For EcoCyc, the production system consists of a computationally translated production rule for every metabolic reaction in the database and a proposition for each known growth nutrient of *E. coli*. Early experimental results are encouraging. The system is able to verify that M63 minimal growth medium is able to produce all 41 essential *E. coli* compounds.

EcoCyc's use of object oriented database technology allows for greater semantic expressivity and resilience to rapid schema changes. The adaptation of expert systems technology for symbolic computing, in Peter Karp's words," represents a distinct nonnumerical style of computing that, allows us to exploit the power of computational qualitative theories" (2001). Yet this implementation of a knowledgebase is programmatically intensive and relies on a suite of add-on tools that are costly to develop and maintain. Since it is based on current expert systems and object oriented technologies, it will be subject to the scalability limitations of its underlying technologies and transformation methodology.

6 ULTRA-STRUCTURE THEORY

Ultra-Structure Theory (UT) was developed by Jeff Long as a theory of systems design in 1985. It aims to optimize computer representation of complex, conditional and changing rules by finding and modeling the unchanging features of changing systems (Long and Denning, 1995). Ultra-Structure has a unique world view of complex systems and processes that distinguishes itself in three main ways:

i. the application of a higher level of abstraction than other systems

ii the rule-centric approach to system organization

iii abstraction of both the system domain (relational DB) the reference (real world) domain

6.1 Conceptual Distinctions

UT is loosely based on the work of linguist Noam Chomsky. In 1957, Chomsky developed an abstract theory of language called the Theory of Transformational Grammar. Chomsky postulates that every intelligible sentence conforms not only to specific grammatical rules of the language in which it is spoken, but also to “deep structures,” which represent a universal grammar that is innate to the human brain and is language independent. Chomsky and other linguists have formulated transformational rules, that transform the grammatical structure of sentences while preserving their semantic meaning. The hypothesis is that to accurately describe the grammar of a human language, you must discover its deep structure. Transformation rules applied to the deep structure transform the surface structure or spoken sentence into immeasurable syntactic combinations. In this manner, transformational rules can describe the set of all the sentences that can possibly be formed in any language (Lechte, 1994)

6.2 Ontological Methodology

6.2.1 The world as a process

Like GO, UT has a process ontology style world view that conceptualizes what a domain does rather than what it is. The aim is to analyze reference domain processes and model them in an evolving prototype fashion until all logically possible processes are represented, not just within the domain of the system being modeled, but in the larger set of systems in which that domain exists. Long argues that like Chomsky's universal deep structure of grammar for all languages, all sets of systems share a deep structure. By studying the processes of domains within a given set, transformation rules can be extracted that eventually lead to the discovery of the deep structure of the entire complex system. The set of all games or all music are examples of complex systems to which UT has been applied.

6.2.2 Rules as process abstractions

Ultra-Structure theorizes that since all processes are governed by rules, as such they can be abstracted to, and represented by first order logical rules in an "If→Then" format. Here, Ultra-Structure applies the same logical rules of inference that are used in ontologies and expert systems to capture the semantic knowledge of complex systems via abstraction. Abstractions manage the complexity of a system by modularizing specific concepts into more general concepts, thereby improving system tractability.

Rules are a more abstract form of description because they are semantically expressive and unambiguous. For example they describe behavior as well as mechanism, and the set of rules explicitly defines the ontology of the system they describe. "If→Then" rules can be used to describe virtually any type of relationship between two or more concepts including user defined relationships. UT uses a specific type of conditional "If→Then" rules. Instead of the typical format of If A is true than decide that B must be true, UT declares that If A is true then consider B before deciding what is true. This is a more expressive form of declaration in that it allows for the consideration of variables that may or may not be relevant to a decision. Variables may be strung together to represent more complex considerations (Long and Denning, 1995). The set of all rules describing a system represent a constituent structure analysis of the reference domain.

6.2.3 Ruleforms as rule abstractions

A scientific law is a set of one or more rules that generalizes the natural behavior of a system. Once a scientific law is accurately represented and tested it will reliably predict phenomena in the system to which it applies. In the domain of thermodynamics for example, the ideal gas law ($PV = nRT$) states that a gas is at thermodynamic equilibrium when the pressure and volume of a gas is equal to the product of the number of moles, the gas constant and temperature. (chemPages, 2002). This law combines the rules of thermodynamics to reliably predicting phenomena in the system (the universe) to which it applies. In this respect scientific laws are an abstraction of a set of one or more rules that modularizes every possible combination of its constituent rules.

Ultra-Structure refers to laws of a reference system as ruleforms. Long argues that to understand complex systems, we must raise them to a level of abstraction above the observable structure, and even above the rules themselves, to the deep structure of ruleforms that govern the rules. Long suggests that, "Rules may be defined in terms of sets, with each set having a specified and limited possible range or set of values. A particular rule in this definition is a particular ordered sequence of these values" (Long and Denning, 1995). For example, a particular rule for the first law of thermodynamics would refer to a specific instance or value for each variable. Hence, like scientific laws, ruleforms are an abstraction of the constituent rules that comprise them.

Ruleforms can be decomposed into an ordered set of underlying domains that Long calls "universals". A universal is a domain or class of entities that represents the single most general level of semantic abstraction of its members. This class includes any subtypes or variations that are defined by individual instances of rules. Universals themselves are defined by a set of constituent rules that declare the existence of domain members. In this manner, constituent rules are abstracted into more complex compound rules, and the set of all applicable compound rules are further abstracted into ruleforms or scientific laws of a system.

Long claims that "Ruleforms are to rules, what ordinary algebra is to arithmetic." They are abstract generalizations of the structural concepts and relationships of a system. Just as algebra uses symbols to abstractly represent numbers in order to generalize the laws of arithmetic, Ultra-Structure uses universals to abstractly represent entities in order to generalize the laws of a system. Long specifies his extension of Chomsky's theory in the "Ruleform Hypothesis", which states:

> Complex system structures are created by not-necessarily complex processes; and these processes are created by the animation of operating rules. Operating rules can be grouped into a small number of classes whose form is prescribed by "ruleforms". While the operating rules of a system change over time, the ruleforms remain constant. A well-designed collection of ruleforms can anticipate all logically possible operating rules that might apply to the system, and constitutes the deep structure of the system (Long and Denning, 1995).

6.2.4 Structural layers

UT represents the reference system in three main levels of abstraction or structure: the surface structure, middle structure and deep structure. The surface structure is composed of the set of all entities, processes, relationships, and observable behaviors that are governed by the rules defined in the system. It is the unabstracted complex view of the actual system as a whole from which UT extracts the rules of the system. The middle structure abstracts the surface structure and expresses that abstraction in the form of the set of all constituent rules that govern the processes or behaviors of the surface structure. Long claims that these rules "generate the surface structure". Finally, the deep structure is comprised of ruleforms, the combinatorial set of ordered universals from which all rules are constructed. According to Long, the deep structure specifies the system's ontology by answering the questions that define the essence of the system, such as (Long, 2001):

- What is common among all systems of type X?
- What is the fundamental nature of type X systems?
- What are the primary processes and entities involved in type X systems?
- What makes systems of type X different from other systems?

Figure 19 illustrates the three structural layers of a UT defined system (Long.2001).

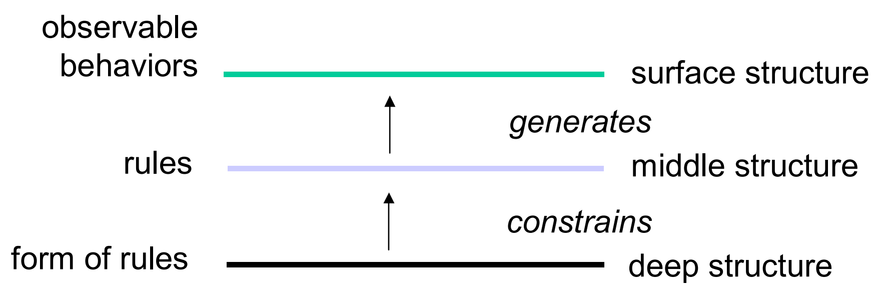


Figure 19: Ultra-Structure structural layers of abstraction

6.2.5 Substructure

A unique abstraction of the UT methodology is the substructure, which consists of the set of universals of a system. This is the abstraction of the particular objects of a system to classes of objects. Object classes represent not just physical objects, but the semantic dimensions that characterize the behavior of a complex system such as location, measurement, time period and relationship. These classes are defined by rules that declare the existence of the class and all of its subtypes. This would normally be the centralized structure that an ontology would subdivide into a type-subtype hierarchy from which a data model would be derived. But in UT this hierarchy is not reflected in the data model. Instead it is encapsulated into a set of complex data types that share global properties and can be manipulated much like objects in an OO database. Assembling universals into ruleforms constitutes a new representation of an upper ontology using a DL like methodology to create rule structure on the fly by forming complex rules from primitive rule components.

6.3 Transformation Methodology

Just as ORDBMS systems are emerging to integrate the best features of relational and object oriented technology, UT theory was developed with the goal of integrating the best features of knowledgebase and database technologies to create large expert systems that are capable of representing extreme complexity. Jeff Long explains,

> A million records is small by database system standards, but a million rules is essentially an impossible number for a traditional expert system

> to manage. We expect to be able to effectively handle very large numbers of rules, numbering in the hundreds of thousands, using Ultra-Structure techniques.

UT was designed to be implemented within a standard relational database system. By disregarding conventional distinctions between algorithms and data and redefining the semantic constraints of relational database design, UT has created a new level of abstraction that overcomes the limitations imposed by conventional design methodologies and tabular database structure.

6.3.1 Abstracting the notational system

Perhaps the most inventive abstraction of UT is that of the notational system. Long is a notational engineer who eloquently postulates why notation is the limiting factor of defining complex systems:

> Abstraction spaces are discoveries, not inventions. We need a more systematic way to develop and settle abstraction spaces. We may have competence in using complex systems but we still don't "understand" complex systems. This is not because of the nature of the systems, but rather because our notational systems – our abstractions -- are inadequate. The notational systems one habitually uses influences the manner in which one perceives his environment: the picture of the universe shifts from notation to notation. Every notational system has limitations: a "complexity barrier". Using the wrong, or too-limited, a notational system is inescapably self-defeating (2001).

UT abstracts the notational system by redefining the semantic framework of relational design to provide a more expressive and modifiable abstraction space.

6.3.2 Redefining semantic constraints

UT redefines the purpose of relational database tables from containers of atomic components of complex entities and their relationships, to containers of the rules themselves. Instead of the relational paradigm, where standard manipulation of component parts must be reassembled and manipulated by the conditional rules of a programming or query language, 99% of the rules of a UT system can be expressed in the data and curated by domain experts. These are the conditional rules of the middle structure, which contains all of the knowledge of the reference domain, expressed as data content. Expressing the rules of a system as data enables domain experts to insert delete and modify the rules of the knowledgebase in a simple native language format that is easy to understand and implement.

6.3.3 Abstracting the syntactic space

UT extends the functionality of the structural or syntactic space of the standard relational database table by imposing a semantic place value constraint to support ruleform order. Place value assigns meaning based on content and location. In the Hindu-Arabic numeral system, place value is column position as a placeholder of 10. In rule forms, place value is the column position as a place holder of the universal and the domain (universal) value of a particular rule. Figure 20 diagrams the UT place value concept.

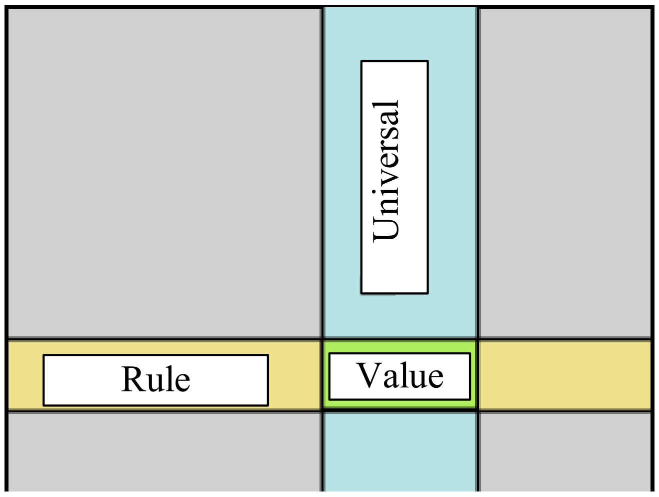


Figure 20: Place value example

Each ruleform is implemented as a table that is designed to model the format of the rules rather than the rules themselves. Table rows are the rules, which are composed of the ordered set of individual attribute values in a ruleform. Attributes values are of two types, factors and considerations. Each ruleform contains at least one factor and one consideration. Factors represent the "If" side of the "If →Then" rule format. They are the conditions by which the considerations should be evaluated. Factors are the unique primary keys for each rule (row), following standard normalization rules. Considerations are the "Then" side of the "If →Then" rule format. They are non-primary key attributes that hold the variables to be considered before a conclusion can be drawn. Figure 21 diagrams a ruleform with factors and considerations.

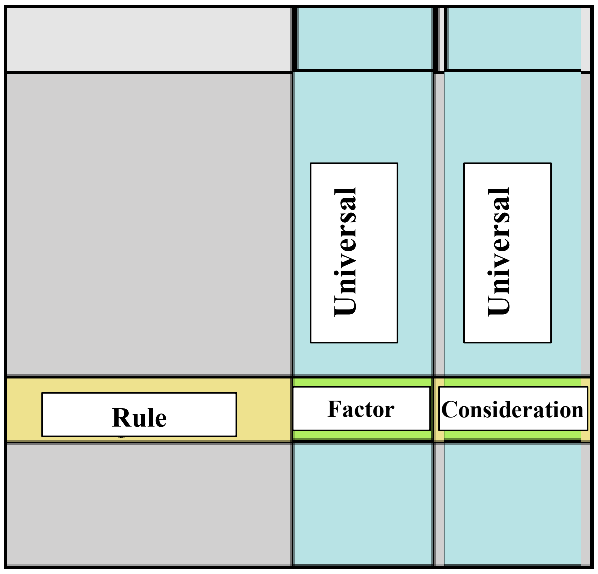


Figure 21: Ruleform example

6.3.4 Rule types and ruleforms

In general, ruleforms model the syntax of all system rules called operating rules. However, there are several subsets of operating rules that take on characteristic ruleforms. The main distinction between types of operating rules is between existential and compound rules. Existential rules declare entities to exist and specify one or more considerations regarding the entities. Minimally, they take an identifier →descriptor ruleform. An example would be name→description. Compound rules relate existential rules by foreign keys (FK), which become compound rule factors and considerations. There are several subtypes of compound rules that have characteristic rule forms.

Network rules relate recursive relationships between entities within the same universal or domain. They typically take the minimal form, "universal1 FK 1, relationship FK, universal1 FK 2". For example, if an existential ruleform declares a universal called "resource" contains two rules, one declaring the existence of a publication and the other an author; and the relationship existential ruleform declares a relationship called is_written_by, then the network rule would be publication FK (e.g. name), (is_written_by) FK, author FK (e.g. name). In this manner the network table relates one "resource" to another similar to the way that RDF triples relate subject, predicate and object.

Other compound rule types include authorization rules, protocol rules and metarules. Authorization rules relate entities from different universals, defining systematic relationships such as part→whole, sequencing and rules of precedence. Protocol rules define related work processes such as conditional steps in a procedure. Finally, metarules, are complex rules for reading other rules.

Metarules have query characteristics in that they are factor value filtering mechanisms. They create custom views or masks of factor value subsets by specifying relationship values. While other rules are clearly reference domain knowledge and as such are expressed entirely in the data, metarules function in a gray area between reference and information system knowledge, hence for efficiency reasons, simple metarules may be hard coded. Since more complex metarules are more transitory, they are expressed as ruleforms. Figure 22 illustrates typical ruleforms for various types of rules.

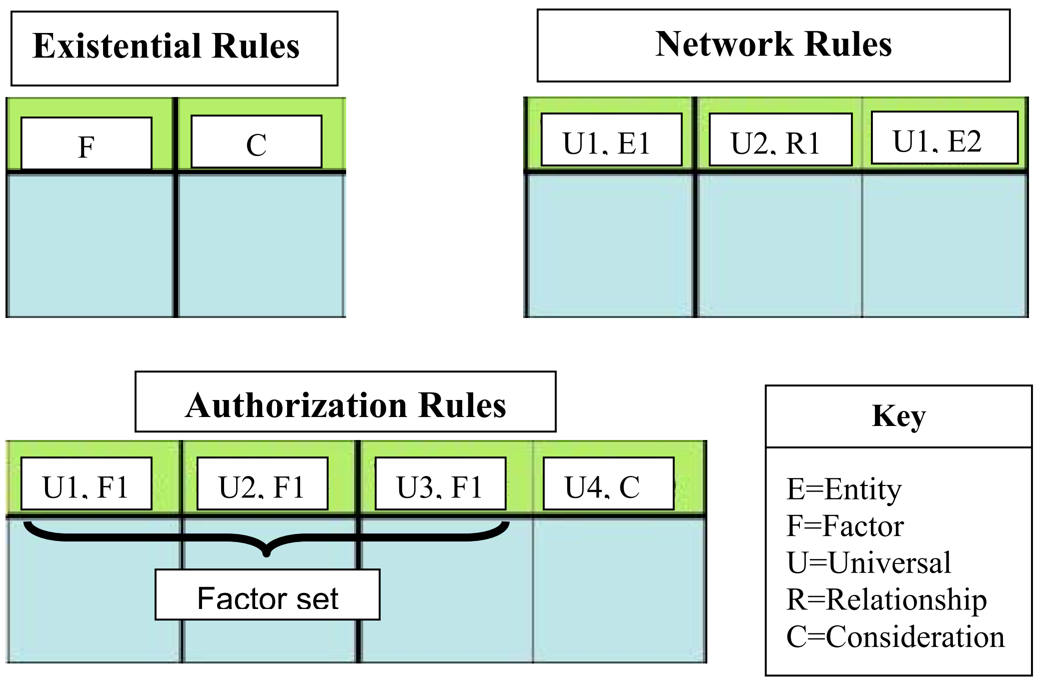


Figure 22: Rule types and ruleforms

6.3.5 Animation procedures

Along with ruleforms, animation procedures comprise the inference engine or "competency rule engine" (CoRE) of a UT system. Although more specific code may be necessary for rare circumstances, in general, animation procedures are rule-form specific routines that act on universal classes of rules within the rule-form. The main advantage of a process-centric approach to abstraction is that the members of a class of objects share common processes and properties and therefore can be manipulated globally with the same programming routine or method. For example, all mammals are warm blooded, all time periods have a beginning and an end, and all measurements have units.

While most programming code is written at the middle (rule) level of abstraction, animation procedures need to code little to no specific rules. Instead animation procedures use the information provided by ruleforms to determine which rules will be read in what order, and which will be executed. In this way, animation procedures resemble OO navigation methods. Essentially, putting the rules in the data rather than in the programming frees the inference engine to define, determine and execute only "what" the system processes should do, the "who, when, where, why, and how" rules are defined in the data. In the "CoRE Hypothesis", UT claims that an incredibly small number of ruleforms and animation procedures can define even the most complex and dynamic of systems (Long and Denning, 1995).

The CoRE Hypothesis

> UT can create “Competency Rule Engines”, or CoREs, consisting of <50 ruleforms, that are sufficient to represent all rules found among systems sharing broad family resemblances, e.g. all corporations. Their definitive deep structure will be permanent, unchanging, and robust for all members of the family, whose differences in manifest structures and behaviors will be represented entirely as differences in operating rules. The animation procedures for each engine will be relatively simple compared to current applications, requiring less than 100,000 lines of code in a third generation language.

Table 5 provides a summary of UT concepts.

Table 5:Ultra-Structure Concepts Summary

Natural Language Term	Ultra-Structure Instance Name	Ultra-Structure Level Name	U-S Implementation
behavior, physical entities and relationships, processes	particular(s)	surface structure	system behavior
rules, laws, constraints, guidelines, rules of thumb	rule(s)	middle structure	data and some software (animation procedures)
(no standard or common term)	ruleform(s)	deep structure	tables
(no standard or common term)	universal(s)	sub-structure	attributes, fields
tokens, signs or symbols	token(s)	notational structure	character set

6.4 Ultra-Structure Systems Design Strategy

Ultra-Structure uses an evolving prototype approach that is similar to that of the Methontology. Ontology development, documentation, ontology transform, system development and knowledge curator training are all produced simultaneously using rapid prototyping. A typical systems design lifecycle for a large scale project involves iterative expansion and refinement of the prototype until the representative set of ruleforms for the domain is defined. This can usually be accomplished within six months of continuous development and with 4-6 versions of the prototype. Documentation consists of a running tally of functional requirements called a function list, and a detailed encyclopedia of concept definitions and design assumptions

Rapid prototyping enables the domain expert to visualize and explore the abstraction space to determine where the system needs to be expanded, while gaining knowledge curator training. Like the Methontology's evolving prototype methodology, UT provides the same intermediate representations that "bridge the gap between how people conceive a domain, and how it is formalized" (Blaquez et. al., 1998).

Discovering the deep structure of a reference domain of any ontology is akin to assembling a jigsaw puzzle, determining where the more obscure pieces fit only becomes apparent in the context of a partially assembled framework. Actively participating in the development process allows the domain expert to directly assemble the puzzle, rather than describe how it should be assembled. This method provides the context for discovery of obscure pieces that are not predictable and would otherwise be overlooked.

6.5 Ultra-Structure Applications

UT has been implemented in several large scale applications including included order-entry through cash-application, financial and cost accounting, inventory and purchasing, equipment and facilities maintenance, executive information, automatic software documentation systems.

6.1.1 Business application

While UT has been implemented in many business applications, One application that handles order entry, inventory control and billing, has been in continuous operation since 1986. This application supported the organization's growth from $13 million to $100 million in annual revenue.

During this time, annual maintenance costs have remained stable at approximately $12,000. Since maintenance is handled by knowledge curators, there is no need to employ any permanent IT staff. Compared to the 1% of revenue standard industry cost estimates, the UT system has been 98% less expensive to maintain.

6.1.2 Document classification application

Document classification System More recently, UT has been implemented by the US Department of Energy (DOE) as an expert system security clearance classifier of government documents. The system includes a large knowledgebase of over 100,000 rules representing a broad range of key document concepts. This program was awarded the "Hammer Award" by Vice President Gore's office in October 1998.

6.1.3 mRNA translation prototype

Experimental UT implementations have been created for legal systems, games, artificial life, music and most recently a biological prototype application that models mRNA translation to polypeptide.

The mRNA prototype, called CoRE 576, has demonstrated functionality as an mRNA to polypeptide translation simulator and sample tracking system. It includes a Network Propagation Engine (NPE) was implemented that performs *modus ponens* deductions based on the protein expression knowledge rules that are expressed as data in the knowledgebase to determine the contra relationship between entities. A simple example of a contra deduction is,

If A = B→Then B = A.

The knowledge model is designed to accommodate all logically possible relationships occurring during translation and can be easily extended to include new levels of organization by simply adding the appropriate rules to the data. Table 6 illustrates entries in a network table relating bioentities.

Table 6: CoRE 576 Network Rules Example

BIOENTITY1	RELCODE	SEQNO	BIOENTITY2
Adenine	AlsoKnownAs	1	A
+H2N	MustHavePart	3	Nitrogen
Acid	IsA	1	Molecule by type
Glycine	MustHavePart	2	Hydrogen
mRNA	Transcribes	1	Nuclear DNA
+HN Molecules	MustBePartOf	1	Histadine Molecules
mRNA-00001	MustHavePart	2	Adenine

The NPE animation procedure, examines the rules in the knowledgebase, deduces where contra relationships exist, generates inferred network rules, and populates the database with them. Figure 23 illustrates the NPE process and Figure 24 is the physical relational diagram for the CoRE 576 prototype.

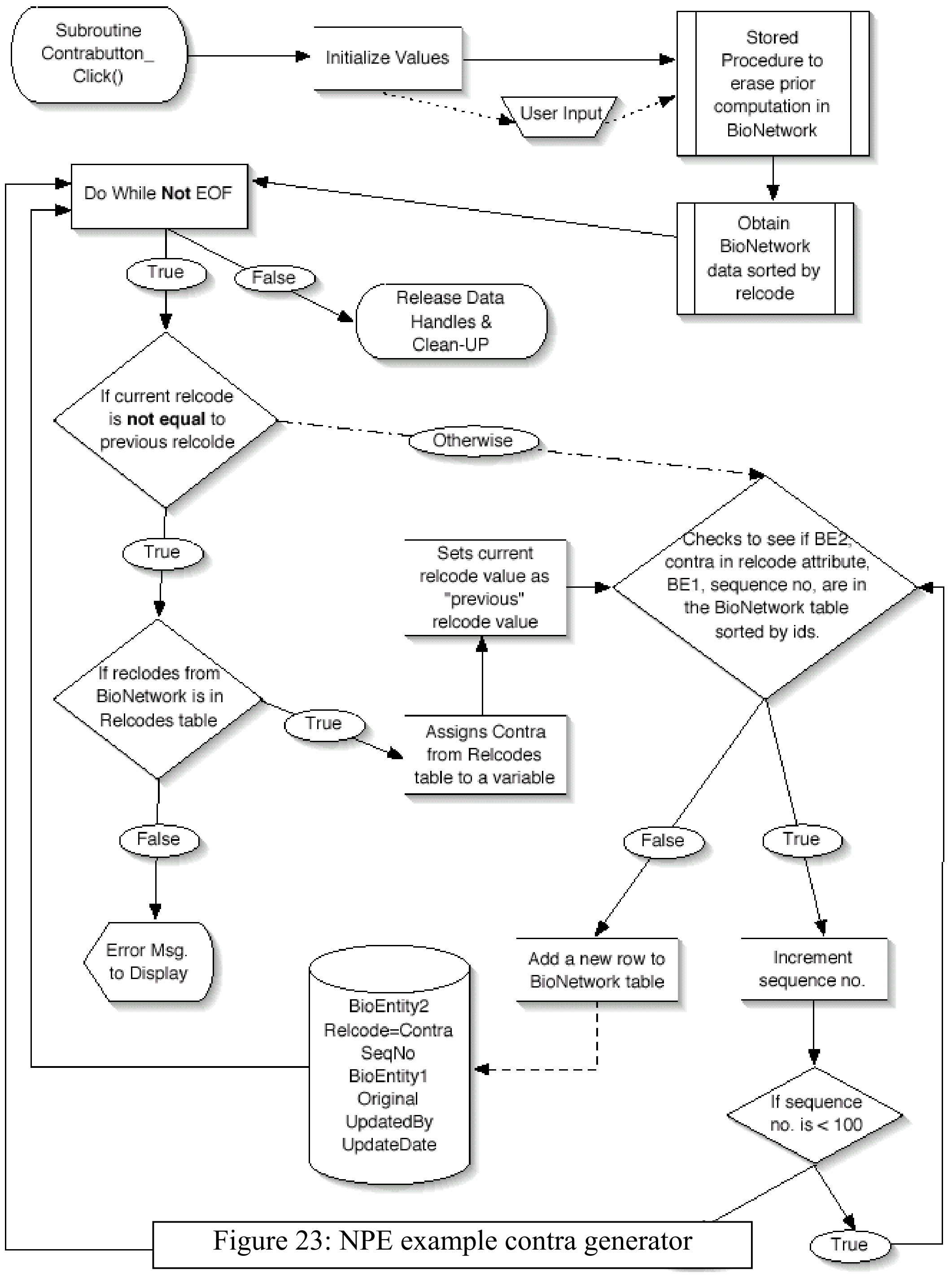

Subroutine Contrabutton_ Click()
Initialize Values
Stored Procedure to erase prior computation in BioNetwork
User Input
Do While Not EOF
Obtain BioNetwork data sorted by relcode
True
False
Release Data Handles & Clean-UP
If current relcode is not equal to previous relcolde
Otherwise
Checks to see if BE2, contra in relcode attribute, BE1, sequence no, are in the BioNetwork table sorted by ids.
True
Sets current relcode value as "previous" relcode value
If reclodes from BioNetwork is in Relcodes table
True
Assigns Contra from Relcodes table to a variable
False
False
True
Error Msg. to Display
BioEntity2 Relcode=Contra SeqNo BioEntity1 Original UpdatedBy UpdateDate
Add a new row to BioNetwork table
Increment sequence no.
If sequence no. is < 100
True

Figure 23: NPE example contra generator

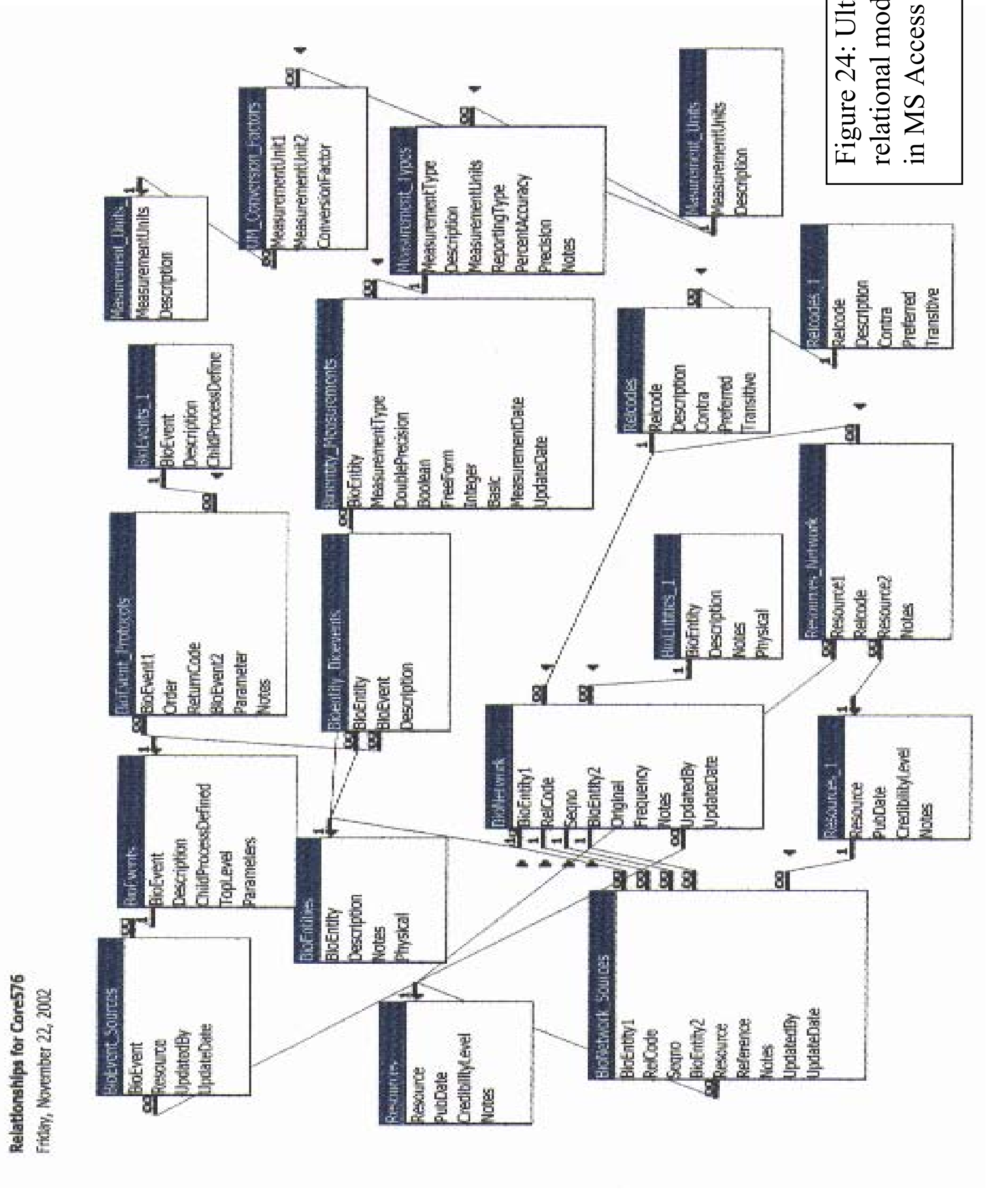


Figure 24: Ultra-Structure relational model in MS Access

7 ANALYSIS

UT approaches the multifaceted challenge of representing dynamic complex systems in a database format with a multidimensional ontology transformation methodology. This section describes how UT addresses the various aspects of the problem, and how UT uses ontological abstractions to create a new level of conceptual encapsulation. This new conceptual level, effectively transforms ontologies into manageable database schemas.

7.1 Alleviating the Critical Bottleneck

The disparity between biological data production and useful data access and analysis is essentially a knowledge processing issue. The inability of current systems design to accommodate rapid changes in the parameters of knowledge processing models is a limiting factor. Instead of encoding logical rules into production systems to manipulate the data, putting the knowledge rules of the reference domain in the data enables domain experts to quickly and directly customize the system by simply changing the data. The result is a high throughput knowledge processing system that is malleable enough to meet the needs of a rapidly changing complex environment in a cost effective manner.

7.2 Overcoming the Cultural Divide

The steep bi-directional learning curve between the two highly complex disciplines of computer science and biology along with the divergent interests of the scientists who study them, has created a knowledge gap that hampers the development of effective knowledge processing systems. UT addresses the cultural divide by separation of system knowledge from reference domain knowledge. UT accomplishes this by abstracting the rules themselves. Since the same encoded methods can manipulate encapsulated classes of objects (universals), these classes can be manipulated like object oriented classes or packages. This frees the programmer to model "what" the system does without needing to know the encapsulated details of the who, where, when, and how.

These details represent the knowledge of the reference domain and are modeled in the data by domain experts. In this manner UT has devised a way to modify relational schemas so that their content can be manipulated in an object oriented fashion. Essentially this enables the programmer to encode in animation procedures, only the rules that manipulate the behavior of complex objects, while the biologist models the detailed knowledge of the reference domain in the data. Hence, each expert is able to model the part of the system about which they are both interested and knowledgeable.

7.3 Reconciling Nomenclature

The UT reference knowledge model is extensible and flexible enough to reconcile innumerable gene name variants via network ruleforms. Domain experts add new synonymous relationships by simply adding new rules in a network ruleform. In this manner, nomenclature variants can be reconciled and gene names aliased on an ongoing basis so that domain experts can gradually build a nomenclature lexicon or controlled vocabulary into the knowledge model.

7.4 Managing Complexity

UT takes a complex theory style holistic approach to analyzing complex systems. UT identifies the classes of universal behavior and the boundaries between them as well as the relevant parameters for affecting system behavior, and then expresses these concepts explicitly in the knowledge management system in the form of logical rules. This methodology introduces a new level of abstraction that captures and represents the knowledge of the reference system, rather than just accumulates data about it. Identifying the universal properties of a system, characterizes the system at a high level of abstraction. This high level view of the reference system enables scientists to understand the system's complexity in the context of an integrated network of perceptual patterns.

7.5 Integrating and Extending Current Systems

Managing the reference system addresses only part of the methodology challenge. Developing a methodology for representing complex systems in an information system really involves the compatibility and integration of two complex systems. The reference system and the knowledge representation system that expresses it. In this case the knowledge representation system consists of database design methodologies and the database itself. The most relevant parameters for description or for affecting the behavior of both systems are system size, complexity, and rate of change. These factors must be compatible between the two systems or an impedance mismatch occurs. Table 6 summarizes these combinations.

Table 6: Svstem Parameters

SYSTEM	SIZE	COMPLEXITY	RATE OF CHANGE
Representational System	Large	Complex	Fast
Real System	Small	Simple	Slow
Representational System	Small	Simple	Slow
Real System	Large	Complex	Fast

Current relational technology such as that of the GO project is optimized for large size, but relatively simple complexity and slow rates of change. OO database technology accommodates complexity and size but not a rapid rate of change. Projects such as the Ecocyc pathways database have clearly demonstrated the benefits of refining ontology-based database projects into expert systems with the ability to perform logical reasoning. Yet, current expert systems technology can handle complexity and change, but does not scale well. UT effectively modifies and extends these conventional systems design methodologies, to incorporate the advantages of each of their technologies.

Instead of developing a static system based on a snapshot of functional requirements, UT creates an environment of perpetual system development that is controlled by the domain expert. The rules that govern system processes can be easily deleted, updated and modified in the data to meet the needs of changing functional requirements. In order to achieve this functionality, UT significantly modifies and redefines conventional systems design methodology in four ways.

First, UT redefines the arbitrary distinctions between programming code and data. From a holistic perspective, the most relevant distinction between programming code and data is with respect to change. The data is easily changed without affecting the structure of the physical system, the code is difficult to change, requires special knowledge of a programming language and often requires interpretation of needs to a programmer in the form of functional requirements. There is no real boundary that precludes defining and manipulating data with data. This allows UT to put most of the production system rules in the data rather than in programming code.

Second, UT redefines the purpose of database applications from capturing and managing collections of data, to capturing and managing the knowledge of domain experts. Essentially, this means changing the goal of conventional database design from optimization for data retrieval, to optimization for knowledge extraction.

Third, UT encapsulates the reference knowledge model into knowledge objects (universals). Discovery of this substructure overcomes the expert systems size barrier by simplifying the data model to a manageable level of complexity for encoded rules, and it translates well within the constraints of a standard relational schema.

Finally, UT abstracts the relational model itself by imposing place value order and representing complex instead of atomic data types. This creates a semantic framework for representing complex data types in the data itself rather than in the structure of the relational model and programming languages.

7.6 Bridging the Ontology Transformation Gap

Ontology-based annotation projects such as GO and EcoCyc have developed large detailed ontologies for representing simple organisms and narrow domains based on biochemical function. These ontologies consist of a logical categorization of object representations and may include processes, behaviors and characteristics. The entities defined within the ontologies are then mapped to database schemas.

The level of abstraction of current ontologies including meta-ontologies is the set of observable objects, and events within the reference domain. This level of abstraction works well for narrow domains and relatively simple systems, yet as complexity increases so does the number of objects and the interactions between them. This is evidenced in the almost non-existent reuse of ontologies for biological domains (Stevens, 2000), the GO's lack of a unifying upper ontology and the fact that MODs of simple organisms such as *E. coli* are extremely complex, time consuming and expensive to develop and maintain. Representing the knowledge associated with a more complex organism will require unifying semantic spaces into manageable high level views that characterize system complexity in the context of an integrated network of perceptual patterns. Discovery of this view requires a higher level of semantic abstraction than current ontologies provide.

UT uses an integrated system of modularized ontologies to capture the semantic knowledge of the reference system and raise it to a new level of abstraction. The knowledge representation (KR) system includes the standard use of ontologies such as those used by GO and Ecocyc to characterize the detailed knowledge of the reference domain. Then UT takes ontologies the next level of abstraction by modeling not just the objects and behaviors of the system, but the logic that governs and predicts them.

While knowledge changes and behaviors vary, the logic underlying them never changes. UT aims to abstract system logic and express it in a rule-based ontology. Instead of applying the rules to the objects in the system UT applies the objects in the system to the rules. It defines a world view of the reference domain in terms of rules of varying specificity. It distinguishes rule types based on the functions that they support, then it maps the rules, and thus the logic that they represent directly to the knowledgebase.

The rules that apply to domain expert or reference system knowledge are expressed explicitly in the data. These rules represent the middle structure and constitute a standard ontology expressed in axioms. The middle structure is semantically integrated using foreign keys and ruleforms rather than a physical hierarchical structure. Eliminating a physical hierarchical representation of objects allows for the grouping of related objects into classes based on the common rules that govern them. These classes become the objects of the highest level of abstraction, the deep structure. This method is similar to Bouaud's concept normalization for fixing the necessary intensional conditions of types, which forms the basic building blocks of the ontology.

The deep structure constitutes a meta-level ontology that represents a new level of logical abstraction. Like Chomsky's transformational grammar, UT abstracts the rules of observable behavior into a network of logical patterns. The patterns capture the underlying logic that governs and predicts those behaviors. This logic is expressed in ruleforms, which are complex rule objects that define rules based on the logical patterns of order of their classes. The ruleform method enables the organization of thousands of rules into very few ruleforms. Figure 25 diagrams the hierarchy of the rule ontology.

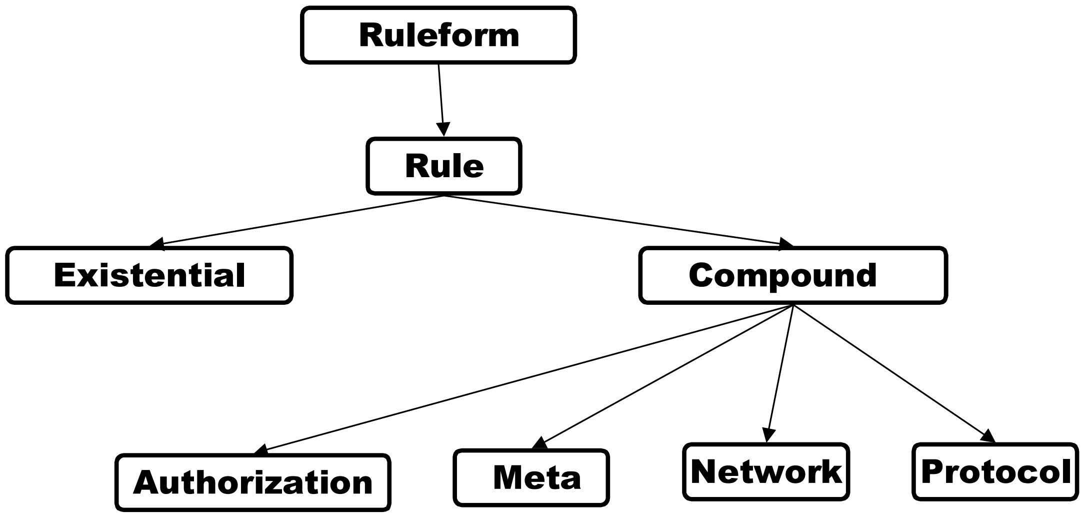


Figure 25: UT Logic Ontology

The metalevel of abstraction reduces the complexity of the global schema to a more compact knowledge representation that it is easily mapped to a standard relational database without sacrificing content. All of the more detailed, more variant and complex relationships are represented in the data as modular local schemas that are integrated by the global schema. This means that more complexity mostly means just larger tables which relational databases are well equipped to accommodate.

Place value constraints offer a third syntactic dimension in which to express semantic constraints within the relational paradigm. Together, the explicit rules, ruleforms and the place value framework constitute an ontology metalanguage that is semantically expressive and syntactically flexible enough to preserve the semantic linkages of an ontology in standard relational database form. This method applies established ontological principles such as Boaud's intensional typing without requiring an elaborate hierarchical structure. The semantic meaning of concepts are dependent upon their intensional distinction from other types and their position in the ruleform rather than in a hierarchy.

8 DISCUSSION

8.1 Potential Challenges and Future Research

Major challenges and areas of future research concentration related to the implementation of UT in biological domains include:

- Ontological issues

- User interface issues
- Scalability issues
- Inferencing issues

8.1.1 Ontology issues

Numerous biological ontologies exist that characterize narrow biological domains. Together these conceptualizations specify a broad range of biological knowledge. Yet they vary widely in scope and granularity (Stevens, 2000). A major challenge in creating a generalized model of biology in a UT system is the integration and extension of these disparate information models into a single global schema. Reconciling lossy and non-lossy clashes related to ontology integration and transformation is likely to be a complicated and time consuming effort. However, once developed, the resulting global schema should be easily maintained by domain experts and general enough to be used as a defacto standard for the annotation of future biological knowledgebases.

8.1.2 User related challenges

The two main user related challenges are the development of an intuitive and efficient ontology editor and the knowledge curator training of domain experts. An ontology editor is a user interface that allows domain experts to effectively interact with the reference ontology. A system of this size and complexity may contain over 100,000 rules that must be accessible to the domain expert in a navigable format. The editor interface will need to group rules into manageable subsets that are intuitive and easy to visualize. This can be accomplished using standard usability testing and interface design methodologies.

Along with the use of the ontology editor interface, domain experts will need to be trained to preserve data integrity by entering data in a consistent manner. The integrity constraints of the underlying database, augmented with data entry constraints in the editor interface and animation procedures can handle most data integrity errors such as circulatory errors where an entity is defined as a subtype of itself and indirect repetition where an entity is defined as a direct subclass of a parent and grandparent entity (e.g. dog as a canine, canine as an animal, dog as an animal) (Stevens, 2000).

However, there is the possibility of entering nonsense data or data that is factually incorrect. Since the knowledgebase assumes that all manually entered is factually correct, therefore incorrect data will produce incorrect inferencing results. The ability of expert systems to produce valid results is wholly dependent on the quality of the data on which its assumptions are based. Knowledge curators must be made aware of this vulnerability and be responsible for ensuring the accuracy of rules entered into the system. The accuracy of data can be validated by tracking associated source or citation data, identity of the person entering the data, and the timestamp information, as well as by assigning authority rankings to the data.

8.1.3 Scalability issues

Although UT systems have been implemented in large commercial domains, none of these applications match the complexity of post-genomic biology. Theoretically, UT should scale to extremely high levels of complexity that encompass the whole of a natural kind or family of systems such as all businesses or all biology. However, this hypothesis has only been partially tested. No UT system to date currently describes all the aspects of a given natural kind of system such as biology.

Unforeseen scalability issues may exist that will only be revealed at high levels of complexity. These may include limits on the knowledgebase systems themselves or on the semantic interoperability mechanisms of UT theory in broad ranging domains. The underlying technology was not intended for supporting complex semantic interoperability networks and may exhibit unforeseen limits in scalability. Although not as likely, similar limitations of the theory mechanisms themselves may emerge. For example the system may reach the point of complexity where there are too many rules to manage effectively. Future research to test the limits of the platform can only be accomplished by implementing the system in conditions of increasing complexity.

8.1.4 Inferencing challenges

The inferencing capabilities of UT systems represent the most challenging and most exciting aspect of future research. Assuming a base knowledge of valid information, UT inferencing may eventually be used for hypothesis testing simulation, visualization and in silico modeling of biology. Future UT systems may perform more sophisticated statistical inferencing for hypothesis testing and knowledge extraction.

8.2 Conclusions

Although UT integrates and extends current methodologies, it is not a new theory. In fact it predates most of the methodologies mentioned in this thesis. All of the ontology methodologies, as well as OO theory were practically nonexistent when UT was developed in 1985. The goal of this thesis is to identify UT as an ontology driven database methodology, to relate the parallel development and theoretical commonalities that exist between UT and more recent ontology driven database methodologies, and to explore UT an ontology transform solution.

UT is an ontology-driven database design methodology that may effectively address the dynamic functional requirements of complex biological systems. By integrating and extending current theoretical and methodological perspectives, UT has developed two key mechanisms that significantly increase its semantic expressivity while providing for rapid change.

First, transferring the part of the production system that represents the reference domain entirely to the data within a standard relational database structure enables biologists to directly customize the system without the need for programming. It empowers biologists to act as knowledgebase curators who are able to quickly and directly adapt the rules governing the reference system as domain knowledge changes. This mechanism gives ultimate flexibility and power to the domain expert and effectively accommodates rapid, perpetual change.

Second, UT expands the world view of relational database design to include place value order and the use of complex objects as attributes, which provides a new syntactic dimension for semantic expressivity. This high-performance, transformation mechanism provides a simple, yet complete semantic bridge for ontology transformation.

The inventor of the modern skyscraper, architect Louis Sullivan coined the term, "Form follows function." Sullivan posited that just as natural forms, such as eyes, ears and wings, reflect specific functions, man made forms should reflect the functional purpose for which they are built. He redirected the world view of architectural design to focus on creating forms that are designed for optimal strength, efficiency and functionality rather than style or beauty. Sullivan's theory was developed to address the increasing demand for larger, stronger, more functional buildings as a result of the industrial age.

As a result of the current technological age, the application of computers is increasing the complexity, as well as the velocity and pervasiveness of change of almost every scientific discipline. Post-genomic biological disciplines, high throughput data acquisition methods, and large-scale science projects has put biology at the forefront of this phenomenon. The rapidly increasing volume and complexity of information being produced by the biological community has created a demand for cost effective knowledge management systems that are flexible enough to accommodate rapid change and semantically expressive enough to capture the knowledge of complex systems in an intuitive format.

Just as Sullivan's methodology resulted in the development of new types of structures (skyscrapers) that reflected the emerging functional requirements of the industrial age, systems design methodologies such as UT are emerging to address the functional requirements of the technological age.

UT lets form follow function by applying well known principles for ontology development such as Bouaud's intensional types without requiring the construction of a complex hierarchy or database schema. By expressing the complexity in data rather than programming code, UT provides a flexible semantically expressive knowledgebase solution using standard relational database technology. The resulting knowledge management systems provide a practical ontology transform mechanism for capturing and managing the knowledge of complex biological systems while accommodating rapid change. UT's rule based approach creates a unique form of knowledge representation that extends the world view of current methodologies to reflect the function of dynamic complex systems.

Acknowledgements

Thanks to Jeff Long and Melissa Kimball for their extensive mentoring efforts to help me understand UT and apply it to real world problems. Thanks to Len Olszewski and John MacMullen for inspiring me, and to my advisor, Brad Hemminger for guiding and tolerating me. Thanks to Chris Goolsby and Eloise Currie for your understanding and encouragement. Finally, special thanks to my husband Howard, without whose patience and care, this paper and in fact this Master's degree would not have been possible.

Figure Legend

References

Bar-Yam, Yaneer (In Press, 2002). *Unifying Principles In Complex Systems*. Online: http://necsi.org/projects/yaneer/ComplexSystems.pdf

Beck, Howard. University of Florida. *Agricultural decision support systems*. Online: http://hammock.ifas.ufl.edu/ABE6644/expertsystems.html

Berners Lee, Tim. August 30, 2001 keynote speech at Software Development East, Boston, MA. In [Alexandra Weber Morales "*Web founder seeks simplicity*" Show Daily Online, 2001] http://www.sdgnews.com/sd2001es_006/sd2001es_006.htm

Beyer, Hugh and Karen Holtzblatt. (1999). *Contextual Design* ACM Press New York, NY, USA PP 32 - 42

Biskup, J. (1998). *Achievements of relational database schema design theory revisited*. In B. Thalheim and L.Libkin, editors, Semantics in Databases, volume LCNS 1358, Springer Verlag, pp. 29-54.

Blake JA, Davisson MT, Eppig JT, Maltais LJ, Povey S, White JA, Womack JE. (1997). *A Report on the International Nomenclature Workshop held May 1997 at The Jackson Laboratory, Bar Harbor, Maine U.S.A.* Genomics 45: 464-468.

Blazquez, M., M. Fernadez, J.M. Garcia-Pinar, and A. Gomez-Perez. *Building Ontologies at the Knowledge Level using the Ontology Design Environment*. In Proceedings of the 11th Knowledge Acquisition Workshop, KAW98, Banff, Canada, April 1998.

Bouaud, J., Bachimont, B., Charlet, J., & Zweigenbaum, P. (1995). *Methodological principles for structuring an "ontology"*. (Report No. DIAM Rapport Interne RI-95-148). Paris: Dpartment Intelligence Artificielle et Mdecine [DIAM]. [Presented at the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, 19 August 1995, Montreal, Canada.].
Online: http://citeseer.nj.nec.com/bouaud95methodological.html

Chandler, Jane and Steve.Hand. (1998). *INTRODUCTION TO OBJECT-ORIENTED DATABASES*. Online:http://www.dis.port.ac.uk/~chandler/OOLectures/database/database.htm - Relational_Model

InContext Enterprises, Inc. (2002). *Contextual Design Diagram* Online:http://www.incent.com/cd/cdhow.html

FOLDOC Free Online Dictionary Of Computing: http://wombat.doc.ic.ac.uk/foldoc/

Fujimura, Joan H. (1995). *A View of Japanese Genome Informatics.* Genome Informatics News, Vol. 2, No. 4.
Online: http://www.genome.ad.jp/documents/news/A9510.html

Guarino, Nicola, Massimiliano Carrara and Pierdaniele Giaretta.(1994). *An Ontology of Meta-Level Categories* LADSEB-CNR Int. Rep. 6/93

HGNC (Human gene nomenclature Committee Homepage). (2001)
Online: http://www.gene.ucl.ac.uk/nomenclature/activities.shtml

Jacob, E.K., & Albrechtsen, H. (1997). *Constructing reality: the role of dialogue in the development of classificatory structures.* In I.C. McIlwaine (Ed.), Knowledge organization for information retrieval: Proceedings of the 6th International Study Conference on Classification Research, 14-16 June 1997, London (pp 42-50). The Hague, Netherlands: International Federation for Documentation.

Jones, D., Bench-Capon, T., & Visser, P. (1998).
Methodologies for ontology development. In IT&KNOWS Conference, XV IFIP World Computer Congress, Budapest, August 1998. Available at: http://citeseer.nj.nec.com/135323.html

Karp, P. D. (2001). *Pathway Databases: A Case Study in Computational Symbolic Theories.* Science Magazine. Volume 293, Number 5537, Issue of 14 Sep 2001, pp. 2040-2044. Online URL: http://www.ai.sri.com/pkarp/pubs/01science.html

Klein, Mark and Abraham Bernstein.(2002). *Searching for services on the semantic web using process ontologies.* In *The Emerging Semantic Web - Selected papers from the first Semantic Web Working Symposium*, Isabel Cruz, S. Decker, J. Euzenat, and D. McGuinness, Eds. Amsterdam: IOS press, 2002, pp. 159-172.
Online: http://www.semanticweb.org/SWWS/program/full/paper2.pdf

Lassila, Ora. Ralph R. Swick.ed.(1999). *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium (W3C).
Online: http://www.w3.org/TR/REC-rdf-syntax/

Lechte, John. (1994). *Fifty Key Contemporary Thinkers.* New York: Routledge
Online adaptation: http://pratt.edu/~arch543p/help/Chomsky.html

Long, J. (1999). *A new notation for representing business and other rules*. In Long, J. (guest editor), Semiotica Special Issue: Notational Engineering, Volume 125-1/3

Long, J., and Denning, D. (1995) *Ultra-Structure: A design theory for complex systems and processes.* In Communications of the ACM

Maedche, A., and S. Staab. (2001). *Comparing ontologies: similarity measures and a comparison study.* (Internal Report No. 408). Karlsruhe, Germany: Institute AIFB, University of Karlsruhe. Online: http://citeseer.nj.nec.com/439130.html

Missikoff, Michele. (2002). *Harmonise: An Ontology-Based Approach for Semantic Interoperability.* In ERCIM News No. 51. Online: http://www.ercim.org/publication/Ercim_News/enw51/missikoff.html

Pearson, Helen (2001). *Biology's name game.* Nature Online: http://www.nature.com/nsu/010607/010607-12.htm

Pragmatic Marketing development. *Pragmatic Product Development Diagram* Online: http://www.pragmaticmarketing.com/seminars/grids.asp

Fernstrom, Christer. *Technology Briefing Report Databases.* RENAISSANCE Esprit Project 22010 Online:http://www.comp.lancs.ac.uk/projects/RenaissanceWeb/project/Acrobat/Databases.pdf

Roos, David S. (2001). *Trying to Swim in a Sea of Data* Bioinformatics Science magazine 291 (5507) 1260
(Online): http://www.sciencemag.org/cgi/content/full/291/5507/1260

Stevens, Robert, Carole A. Goble and Sean Bechhofer (2000). *Ontology-based Knowledge Representation for Bioinformatics.* Department of Computer Science and School of Biological Sciences, University of Manchester (online).
Online: http://www.cs.man.ac.uk/~stevensr/onto

Stevens, Robert (2002), ontology final presentation Parts I and II
Online:http://www.cs.man.ac.uk/~carole/old/GGF/FinalTutorial/GGFpart2.ppt

Smith, Mark. (2002). *Strategic Assessment Guide: DecisionCycle Methodology.* Intelligent Enterprise. Figures 1 and 4. Online: http://www.intelligententerprise.com/online_only/saguide2002/saguide.shtml

Sowa, John. (2001) Upper level ontology diagram
Online:http://users.bestweb.net/~sowa/ontology/toplevel.htm

Taylor, Mark C. (2001). *The Moment of Complexity: Emerging Network Culture* University of Chicago Press. Excerpt Online: http://www.fathom.com/feature/122486

Whatis.com Online computer science reference
http://searchvb.techtarget.com/sDefinition/0,,sid8_gci211545,00.html

White JA, Apweiler R, Blake JA, Eppig JT, Maltais LJ, Povey S. (1999). *Report of the Second International Nomenclature Workshop (INW2).* Genomics 62: 320-323.

Yang, Zijiang. (2002). *Using and Designing Database Systems.* Course notes. Online: http://aml.yorku.ca/~zyang/3220a/3220A1.ppt