# CHAPTER FIVE

# HISTORY AND NATURE OF ONTOLOGY

## 5.0 Introduction

The discussion in Chapter 4 highlighted many contentious issues that are still grey areas in the software development field. Among the problems still facing developers is the need to maintain the characteristics of the business and domain models of systems, from the analysis stage to the design and implementation stages (*See Section 4.3.1*). Furthermore, it has been stated that, in order to avoid prescriptiveness, the requirements specification model should be captured and documented in a domain-related language.

This chapter introduces the information systems ontology concept, an artefact that has been undergoing continuous development for roughly the past two decades or so. The term ontology has been used in IS in the early 1990s by Tom Gruber and his research team (Neches *et al.,* 1991; Gruber, 1993), when he was working at Stanford University. Its use in computer science, information systems (IS) and information technology (IT) has been increasing gradually. In recent times it has been used in the fields of Semantic Web and database systems (Gruber, 2008), library sciences (Ding & Foo, 2002), routing systems (Winter & Tomko, 2006), information systems and software development (Mavetera, 2007; Aβmann *et al.,* 2006; Dristas *et al.,* 2005; Corcho *et al.,* 2006; Wand & Weber, 1993; 2002), to mention but a few. Ontology has also been used to access legacy resources (Simonov *et al.,* 2004) through an ontology-driven natural language web-based access system. More interestingly, Soffer *et al.* (2001) used ontologies to bridge the gap between business requirements and off-the-shelf information systems capabilities in order to adapt the business to the software capabilities. As is evident from these few examples, there are many fields that already use ontologies and software development is but one specific application area.

The purpose of this discussion is to outline explicitly the characteristics of ontologies that can be used in software development and, most importantly, to support the objective of this thesis of developing a methodological approach to software development that is ontology-driven. It is hypothesized that an ontological approach to software development can only be achieved if the current gap in traditional software approaches is closed by introducing ontologies in the software development process. The problems highlighted in the traditional approaches are

mainly the result of the lack of approaches that can address the cultural, contextual and finally, the humanist requirements of information systems, as discussed in Chapter 4.

As discussed earlier, this chapter, which focuses on the nature, characteristics and application areas of ontologies, will build a theoretical grounding that will be used to motivate the inclusion of ontologies in software development. Their inclusion may occur at several levels, such as the development or runtime stages of the software products. It may also occur at the knowledge base or process levels (Haller & Oren, 2006). This discussion starts by looking briefly at the history of ontology from its philosophical definition in metaphysics and its subsequent incorporation in information systems. After discussion of the characteristics of ontologies, the various types of these ontologies applicable to this research are also discussed.

To provide a good grounding for its use in the development of software products, the discussion continues by focusing on the methodological and architectural ontology perspective. Finally, a working definition of ontology is discussed, together with that of an ontology-driven framework of components that can be adopted for use in software development. This framework, together with the information systems issues discussed in chapter 4 is used in Chapter 6 to develop an ontology-driven software development approach.

## 5.1 What is Ontology? A Brief Description

*"Ontology makes knowledge visible and accessible and enables teams to share their knowledge and profit from experience."*

***Sheryl Torr-Brown (In-PharmaTechnologist.com, 2005)***

Ontology is a word that originated from classical philosophy as a branch of meta-physics. It is referred to as the science of being (lower-case 'b') (Ruiz & Hilera, 2006) or as the study of existence (Hacking, 2002). As a study of essence, ontology started as a way of categorizing things and establishing the nature of their existence (Corcho *et al*., 2006). It also deals with issues such as how people perceive the world and with general issues of the nature of things as opposed to specific theories about particular things. Checkland (1999) holds that it is a concept that deals with the nature of the world or with what it contains. This definition does not look at the individual fragments of existence but at the general. As Hacking (2002:2) argues, ontology constitutes the thought study of "What there is." It should be noted that this term, 'Ontology' written with a capital letter 'O' has an uncountable reading (Guarino, 1998). A typical example is the statement '*Ontology is the study of existence'* which, in this context,

refers to a specific discipline of study. The philosophical ontology is "neither reducible to, nor identical with language or its formalism" (Zúñiga, 2001:188). However, the language can be used to describe this ontology.

In its use in information systems, Guarino (1998) describes the concept "ontology" (with a lower-case 'o') as having a countable reading. As such, people can refer to ontologies if they talk of more than one type (kind) of a thing. In the context described by Hacking (2002:2), this looks at what is there to individuate and characterize, in other words at, "the particulars that fall under them". An example of such a countable reading is the statement:

*Surgical ontology and pharmaceutical ontology are both examples of medical ontologies.*

Hacking (2002:1) stresses the need to contrast the study of 'being' (lower-case 'b') with the 'Being' (upper case 'B') that is used to individuate entities. In addition, Hacking (2002:2) suggested that people should think of ontology in general as the "What is there that we can individuate?" This view includes both physical and abstract concepts, such as classes, types of entities and even conceptual ideas that people may have about a thing. Ruiz and Hilera (2006) regard even mathematical objects and imaginary things as 'Beings', regardless of whether they are fictitious or unreal. This type of ontology takes as its epistemological stance the notion that objects may be both physical and abstract at the same time or may be natural or created by the action of man (the artificial).

Adding to this, Corcho *et al.* (2006:4) distinguished between 'an ontology', which is a classification of things, and 'Ontology,' which is a branch of philosophy. They argue that while the philosophical 'ontology' may not be computer-processable, the ontology used in information systems should be "codified in a machine interpretable language, that is, the human perspective of ontology should be changed to the computer perspective".

Therefore, ontology (with a lower-case 'o') can also be viewed as an engineering artefact, as a product of design science (Hevner *et al.,* 2004; Peffers *et al.,* 2008). Guarino (1998) contends that, in its engineering form, ontology consists of a specific vocabulary used to describe a certain reality or what Shanks *et al*. (2003:56) called "theories about the structure and behaviour of the real world in general". These are contrasted to subject specific ontologies such as domain and process ontologies. The vocabulary used is accompanied by a set of explicit assumptions, which give the intended meaning of such a vocabulary. In it, certain assumptions have to be made that govern the ontology in its application domain.

These assumptions, however, are expressed in the form of first-order logical theory. The vocabulary contains binary and unary predicate names, which are called relations and concepts respectively (Sowa, 2000).

In systems development, ontology is, therefore, a "representation of knowledge based on objects, concepts and entities existing within the studied area, as well as the relationships existing among them" (Ruiz & Hilera, 2006:50). The purpose of ontologies is to provide machine processable semantic information that can be shared by organizational actors such as humans and software agents (Simonov *et al.,* 2004; Hofferer, 2007).

In-PharmaTechnologist.com (2005) further characterizes ontology as a branch of applied science that deals with the development and use of knowledge networks. They also regard it as a science of context and communication. This brief on the essence of ontology enables us to appreciate the different application areas in which ontologies can be used.

## 5.2 Background to Ontology Applications

Ontologies have found various application areas in the modelling of static knowledge and the semantic Web, in which they define, carry and share knowledge on the Web (Corcho *et al.,* 2006:2). In fact, ontology is used in almost all scientific disciplines (Haller & Oren, 2006; Taniar & Rahayu, 2006; Calero, Ruiz & Piattini, 2006) and thus much research is being done in this field. The scientific disciplines in which ontologies have found some application include knowledge engineering, information engineering, agent-based information systems, knowledge management and artificial intelligence, to name but a few.

Many knowledge-based problems found in information management application, whether intra-, inter- or extra-organizational, can be exploited using ontologies. These activities, as noted in In-PharmaTechnologist.com (2005), include enhanced information retrieval, text mining, annotation of databases (Gruber, 2008), data-mining and the development of tools to facilitate data-discovery and decision-making. According to Ruiz and Hilera (2006:51), in software development, ontologies assist in avoiding "problems and errors at all stages of the software product life cycle". Of particular importance is their role at the requirements analysis stage, where they carry the domain knowledge of the discipline into the software product itself and subsequently to the maintenance stage. Ruiz and Hilera (2006) add that, at the maintenance stage, ontologies enhance the understanding of requests for modification and of the maintained system.

## 5.3 Other Views of Ontology

In addition to Guarino's (1998) definition, Gruber (1993) and Studer (1998) describe ontology as an explicit formal specification of a shared conceptualization. Formalization in the ontology deals with machine readability and conformance (syntactics) to specific standards (Mavetera, 2004b). Explicit specification incorporates the clear identification of concepts, their properties and relations, the functions, constraints and axioms (semantics) within a universe of discourse. Explicitness, therefore, refers to the clarity of its existence and meaning to all people involved in the subject matter.

In discussing this further, it should be noted that, if something is clear to a subject, it should be "sense-making" to that subject (Mavetera, 2004b). The addition of the phrase "shared conceptualization" implies the existence of at least two subjects who share a common inter-subjective world view. This brings in the notions of consensus and agreement on the use of concepts that make up the ontology and their meaning. Furthermore, there should be some sense of mutual understanding of the concepts among people, as well as of how they are applied in that same contextual environment (pragmatics).

To explain the concept of shared conceptualization further, Figure 5.1 shows actors A and B, both of whom have different world views.

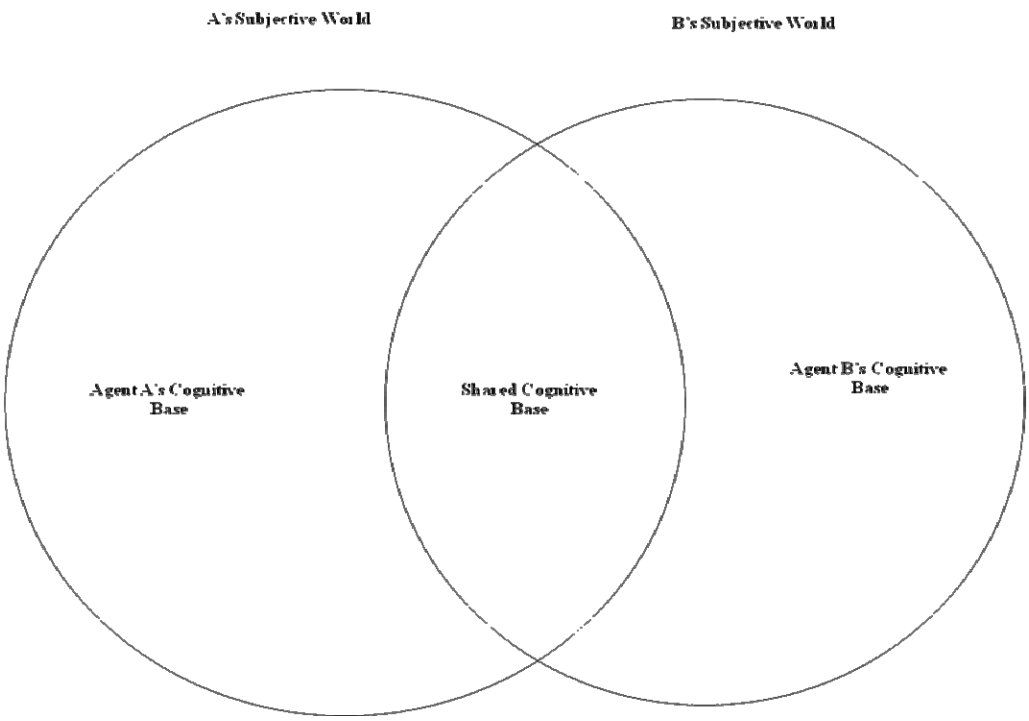


**Figure 5.1: Different Realms of the World** ***(Adapted from Goldkuhl, 2002)***

Figure 5.1 shows that each actor possesses its own view of the world and that this world view can be likened to the actor's ontology. The world view resides in the actors' individual cognitive base. Actors A and B should, therefore, have a shared understanding of their cognitive bases if they are to communicate effectively. This shared cognitive base is depicted as the intersection of the cognitive bases of actors A and B, as illustrated in Figure 5.1. To summarize, the shared cognitive base is, therefore, synonymous with the shared conceptualization of a topic area that is heavily dependent on the context of the situation. This shared cognitive base resembles the ontology of the two actors in a particular domain. This ontology characteristic is very important in organizations when attempts are made to capture organizational context.

At a grammar level, Neches *et al.* (1991) and Uschold and Gruninger (2004) define ontology as the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining the terms and their relations so as to define extensions to this vocabulary. The meanings of these terms should (ideally) be grounded in some form of logic. In agreement with this definition, Swartout *et al.* (1997) and Aβmann *et al.* (2006) also describe ontologies as groups or collections of hierarchically structured concepts that are abstracted from a certain domain. This definition views ontologies as groups of facts that can be used as a skeletal foundation for a knowledge base (Mavetera 2004b), that is, it is a set of representational terms in the universe of discourse. This argument equates ontologies to a group of concepts as defined by Dahlbom and Mathiassen (1995) but not as packages of information.

In another discussion, Dristas *et al. (*2005:6) describe ontology as an "attempt to express an exhaustive conceptual scheme within a given domain". They argue that ontology can be used as a good basis for information modelling. In other words, it is an artefact that is well placed to capture the information content of an organization. Thus, contrary to the earlier discussion that does not accept ontologies as being packages of information, the use of ontologies in information modelling can be used to support Mavetera's (2004b) thesis that ontologies can be viewed as information packets. However, Aβmann *et al.* (2006) advocate the view and use of ontologies as models. In this form, ontology becomes a model shared by a group of people in a certain field. Unlike ordinary models, ontologies need to be shared and should adhere to the open world assumption principle. The open world assumption principle holds that

anything that is not explicitly expressed by ontology is not refuted or negated but is regarded as a fact that is not available and, therefore, not known.

In utilizing the ontology artefact, different researchers have varied perspectives, depending on the proposed outcomes. There are, however, two prominent perspectives that are currently gaining ground in information systems: the methodological and the architectural views of the ontology artefact.

## 5.4 Ontology Perspectives

The methodological perspective focuses on a very high interdisciplinary approach. In this approach, the role of philosophy and linguistics in analyzing the structure of reality is discussed, the intention being to get a clear and unambiguous vocabulary that is used in a particular domain. This thesis does not discuss this perspective in great detail but focuses on the architectural or structural perspective of ontologies.

### 5.4.1 Architectural Characteristics of Ontology

Architecturally, ontologies consist of classes, relations, axioms and instances (Neches *et al.,* 1991; Aβmann *et al.,* 2006). These ontology classes represent concepts in a domain area (Corcho *et al.,* 2006) and are usually taxonomies of these concepts. Likened to the resource description framework (RDF) schema, these classes may be regarded as resources. Resource description framework is a framework for information representation on the web (W3C). The resource triplet, therefore, includes the subject (Analysis technique), the object (ERD/DFD) and the predicate (the IS_A) relationship, as depicted in Figure 5.2.

These relationships, as in the entity relationship diagrams (ERD), represent associations amongst the different concepts (classes) in the ontology. The next component of ontologies consists of the formal axioms. These axioms are used for modelling knowledge, that is, for checking the consistency of knowledge in a knowledge base or for inferring new knowledge in that base (Corcho *et al.,* 2006). Lastly, ontology instances are used to represent elements or individuals in ontology. These are instanced from the ontology classes, for example, an ontology class ‘human being’ may have men or women as instances or more specifically, John or Jane, depending on the level of granularity.

Structurally, ontologies are used to represent and describe concepts hierarchically in a domain field (Aβmann *et al.*, 2006). They are computationally independent and may be

likened to the computationally independent model (CIM) described in Section 4.3.1. Although they argue that the structure of ontology may be conceived as a collection or group of concepts that are related through associations (relationships), Gonzalez-Perez and Henderson-Sellers (2006) add that the primary purpose of these concepts and, hence, of the ontology is to be able to be used.

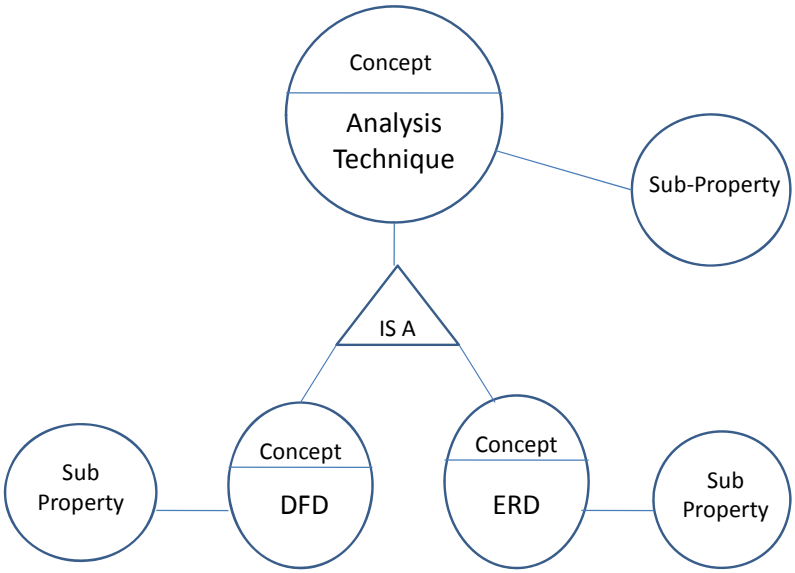


**Figure 5.2: Structural Representation of Ontology**

Figure 5.2 illustrates three concepts: analysis technique, data flow diagram (DFD) and entity relationship diagram (ERD), all of which are included in the discipline of systems analysis and design. Analysis technique is at a higher level (generalization) than either the DFD or ERD techniques, which are specializations of the analysis technique. The IS A relationship is defined as the association between the three concepts. In addition, each concept is associated with at least one sub-property whose purpose is to fit the concept in a specific domain. It is not surprising that each concept when used in different domains may resultantly have different sub-properties. These sub-properties can also be regarded as concepts and are very important for capturing context.

Each of these concepts is an intensional definition of a set of potential entities and associations between any two or more concepts in a formalization of a group of potential relationships between these concepts. Since the concept has to serve some purpose, the sub-properties of the concepts, being entities that possess the definitions of these concepts, are

also included. These sub-properties complement the semantics in the concepts with pragmatic information. This pragmatic information is subject to change as and when new concepts are created and added to the existing knowledge base of the domain. Any category is thus subject to change, depending on the nature and context of the domain.

## 5.5 The Concept of Conceptualization

*"Formal languages such as logic and computer programs are precise, but what they express so precisely may have no relationship to what the author intended."*

***Sowa, 2006.***

Authors of prescriptions are always targeting the real world as the item to be mapped. However, the real world has so many states that they cannot all be represented in formalizations such as software products. The gap between what is real and desired and what can be represented can be reduced if real world extensional relations are made to be congruent with the intensional models of software systems. Checkland (1999) argues that the extensional variety of human systems cannot be exhaustively mapped into the conceptual model and that a selection has to be made on what can be modelled and later implemented. This requires analysts to agree on a shared system conceptualization that can be implemented in practice. This conceptualization relies on the ability of developers to create an intensional model of the system. This is all based on what is called a conceptualization. Most of the practical relevance of ontology in information systems results from its definition as a shared conceptualization. This section briefly describes a formal model of a conceptualization.

Using Guarino's (1998) formal characterization of a conceptualization, a domain space <D, W> is defined in which D is the domain and W is a set of maximal states of affairs of D. W can also be referred to as a set of possible worlds or as the requisite variety of a system. A conceptualization, <D, R> is thus a structure in which D is still the domain and R is a set of relations, which can be applied in D. Unlike the mathematical definition of conceptualization that emphasizes extensional relations, that is, a description of what an outsider can perceive about the system, Guarino (1998) preferred people to use an intensional account of conceptualization.

Guarino (1998) characterizes an extensional account of a conceptualization as a set in which all elements in a domain are listed explicitly and which can be regarded as a 'definition by extension'. In terms of possible system states, this definition by extension comprises all

possible system states in the real world. In contrast to this, a specification that defines the property that should be true of each element, the intensional account of a conceptualization, is called definition by intension.

In any system, the intensional relations focus on the meaning of the relationships R as they are applied in D as the domain. These relationships are called conceptual relationships and are defined for relationships in a domain space. In other words, intensional relations are specifications that define the property that each element should possess. Intensional models are, therefore, used to describe what the system should do. Buitelaar *et al.* (2005:1-2) state that the intensional aspects of a system can be used to define and formalize the domain ontology of the system. This assumption is based on the fact that each system possesses application areas or has got intensions. It should be noted that, even if intensional models do not capture all the possible states of a system, they maintain the domain relevance of a system. The following section discusses the different varieties of ontologies that can be used in software development.

## 5.6 Varieties of Ontology

As discussed in Section 5.5 above, ontologies can be referred to as domain-related, intensional models. There are, however, different ways of classifying these ontologies. These can be classified according to their granularity (Guarino, 1998), functions or form (Fensel, 2004), type of conceptualization structure (Van Heijst *et al.,* 1997) or the nature of existence which Jurisica *et al.* (1999) termed the nature of the actual world issue that needs modelling. In their discussion Ruiz and Hilera (2006) noted the existence of another classification criterion that considers or combines two or more of the abovementioned categories. The bi-dimensional classification proposal as they called it depends on the degree of richness of the internal structure of the ontology and the "subject of the conceptualization" (Ruiz & Hilera, 2006:55). In other words, it is based on the architectural perspective of the ontology, as discussed in Section 5.4.

Since the purpose of ontologies is to capture organizational semantics, Hofferer (2007) uses the semantic expressiveness to group ontologies as having weak or strong semantics. WorldNet is an example of ontology with weak semantics, whereas domain specific ontologies, in which the semantic expressiveness is more than that in WorldNet, may be regarded as having strong semantics. It should, however, be borne in mind that this categorization as *weak* or *strong* is quite relative.

This research is not limited to a single ontology classification criterion, as proposed by these other researchers, but uses these different classifications to arrive at a grouping that suits the requirements of this research, that is, the application of ontologies in the software development field.

As software development is a domain in its own right, domain ontologies have to be regarded first. The research will, therefore, concentrate on domain ontologies, method ontologies and on status ontologies that combine static and dynamic ontologies (as proposed by Jurisica *et al.* (1999), on intentional ontologies, social ontologies and, finally, on process ontologies.

### 5.6.1 Domain Ontologies

In any study discipline or domain, generic or specific knowledge has to be captured. This usually depends on the developers' requirements. Domain ontologies, as they are known, capture knowledge that is valid for a particular type of domain. Domain ontologies can also be viewed as formal descriptions of the classes of concepts and their interrelationships that describe a specific field of application. As these ontologies have to be reusable, a knowledge base for concepts related to a domain and their relationships should be developed.

During information systems modelling, domain ontologies can capture the domain model and the business model of the system. However, they cannot capture the system requirements since these are prescriptions of the specific system to be developed (Aβmann *et al.*, 2006:266). In effect, domain ontologies play the role of "standardized analysis models" less the specification model. This supports the notion that all ontologies are developed for the sake of capturing information or knowledge that has to be shared within a certain universe of discourse.

If this classification is accepted, neither the internal structure of a knowledge area nor conceptualization need concern the people tasked with sharing knowledge within a specific field. This is because, depending on their requirements, they naturally have to start at the highest level of the structure and work down to the lowest level or vice-versa. Hence the formality and granularity of the domain ontology can be neglected as a classification.

In Section 4.3.1, the analysis model was portrayed as comprising the domain, business and requirements models. This analysis model is a descriptive type of model that conforms to the open-world assumption. During software development the domain ontologies can thus be used to model the domain and business models.

### 5.6.2 Method Ontologies

Method ontologies capture the knowledge and reasoning needed in the performance of a task. These can be reduced to task or activity ontologies. The basic tenet, therefore, is the 'what?' and 'how?' of doing a particular task, for example, the requirements-gathering task or the software-design task. Task ontologies, consisting of lexical, conceptual and symbolic aspects are used to specify the objects and their interrelationships that are used in the execution of a certain task (De Oliveira *et al.,* 2006). In short, task ontologies are used to describe a task conceptually.

Although De Oliveira *et al.* (2006) went on to give very sound descriptions of the types of tasks and of their components, it is nevertheless of the utmost importance to derive concepts by which the task ontology is formed from the domain ontology. This process reduces the risk of losing the descriptive nature of the ontology in a domain.

### 5.6.3 Status Ontologies

In software development we recognize the need to capture and represent the static or the dynamic characteristics of a system. There should, therefore, be both static and dynamic ontologies. These ontologies represent the status of an artefact. Status ontologies, as they are called here, argue for a world containing artefacts that exist and that do not change their form of existence (static), as well as for another class of things that change with time (dynamic). Dynamic ontologies can thus be used to abstract the behavioural characteristics of a system. These characteristics consist of concepts and their interrelationships (Aβmann *et al.,* 2006). As each software development process should allow the modelling of both static and dynamic states of a system, status ontologies are therefore required.

### 5.6.4 Intentional Ontologies

It must be noted that organizational systems are intentional and their intentions are reflected as objectives (Hirschheim et al., 1995). There must be a way of capturing these intentions in the IS system. This task is given to intentional ontologies. These ontologies are intended to model the softer aspects of living things, of the kinds of things that can have beliefs, desires and intentions. In this category, the human aspects of living things are modelled. Examples of such ontologies are aspect, object, agent and support ontologies, as stated by Ruiz and Hilera (2006). These types of ontologies are also meant to model ascriptions of intentions to actors in a system, as expounded in the discussion of the theory of organized activity *(Section*

*4.6.4).* Yu (1995) discussed the i* framework that position organizational actors as having intentional characteristics that is, beliefs, goals, abilities and commitments. This framework emphasizes the importance of actor dependencies in an organizational system. The i* framework can also be supported with domain and task ontologies as discussed by Mavetera and Kroeze (2010).

### 5.6.5 Social Ontologies

Social ontologies describe the organizational structure and interdependencies existing among the social actors in these organizations. At a high level of abstraction they include concepts such as the roles and responsibility of the actors, to mention but a few. To enable the social nature of organization systems to be understood, three theories were used in Section 4.6: activity theory (AT), actor network theory (ANT) and the theory of organized activity (TOA) to expand on this. In software development, actor roles, such as those of analyst, developer, tester, to mention but a few, can be used as examples.

### 5.6.6 Process Ontologies

As Haller and Oren (2006) noted, process modelling activity has been used to describe the intended dynamic behaviour of organizations, as well as to capture the processes and context of these organizations. The organizational context includes data, the roles played in the organization and the resources that are utilized in these processes.

In most system development projects, requirements specify the processes to be used in these systems. These processes, captured as requirements, are usually modelled using the specification model. The specification model is very prescriptive in nature and hence conforms to the closed world assumption. It cannot, therefore, capture the intended system behaviour.

However, process ontologies can capture the three aspects of enterprise knowledge, i.e., context, content and structure. These are reflected as the enterprise knowledge, the domain knowledge and the information knowledge respectively. Enterprise knowledge consists of processes, organizational structure, IT structure, products and customers. Domain knowledge comprises the terms, concepts and their relationships and, lastly, information knowledge consists of the types of documents and document structures that are used in the organization. Process ontologies can capture this triplet as a meta-knowledge model for the organization (Haller & Oren, 2006). Hence, the addition of process ontologies as a specification model

may remove the closed world assumption nature of the specification model. *(See Section 4.3.1).*

Process ontologies and method ontologies can be regarded as fulfilling the same role in software development. Also worthy of note is the fact that a process repository can be used to capture process models, business rules and checklists, application data and document knowledge and the background knowledge. It should be noted that several ways can be used to classify ontologies. The present classification is not exhaustive and is only included to complement the requirements of this research study.

## 5.7 Possible Ontology Uses in Software Development

There are several fields in which ontologies can be applied. In this section, we again concentrate on those ontology characteristics that are perceived as being important in software engineering or development. Many software development practices decry the lack of sound communication methods or techniques. The discussion will thus focus on those ontology characteristics that can enhance communication during this process. In addition, modelling, system integration, database integration and software reusability will be discussed. Of all the ontology qualities, communication is regarded as the greatest contribution that ontologies can make to the field of software development.

### 5.7.1 Ontologies and Communication

In their paper Gonzalez-Perez and Henderson-Sellers (2006) advocate the use of a common vocabulary or conceptual base that can enhance communication amongst stakeholders during software development. They hold that, as an explicit account of a shared conceptualization, ontologies reduce the conceptual and terminological ambiguity of terms among software developers. They also provide a shared cognitive base that can be used in any domain. Through the creation of an ontology-driven knowledge base, the contextual understanding amongst varying software engineering stakeholders can be captured, stored and shared. Aβmann *et al.* (2006) support this, saying that ontologies provide a common (uniform) communication language between the software architect, customer and the domain expert. This increases understanding and interoperability among stakeholders.

The need for an ontology-driven knowledge base becomes critical in environments where software outsourcing is prominent. In these environments, stakeholders in different countries or continents can use the ontology base to share domain knowledge, process knowledge and

task knowledge electronically. An example of this is a case in which an analyst is based in the Americas, a developer in India and the users in Africa.

### 5.7.2 Ontologies in Modelling

Ontologies have been characterized as models. As models, they can be used in systems modelling. In an attempt to develop an analysis model, ontologies can be used to create domain models, business models and specification models. In such cases they have to be translated to represent a specific group of systems in the domain. Many organizational systems are usually represented using system models. Aβmann *et al. (*2006:254) characterize system models as "models that describe or control a set of systems". System models should, therefore, comprise both the structural and behavioural models of the system. The role of ontologies in modelling is further supported by the work of Shanks et al. (2003), Wand and Weber (1993, 2002) who used ontologies in conceptual modelling and also Guizzardi and Halpin (2008:2) who used them in evaluating "conceptual modelling frameworks". However, Wand and Weber (2002) urged practitioners to make an empirical determination of the importance of ontologies in software development. In other words, ontologies must be used in an application area to judge their usefulness.

As described on the section on status ontologies (*Section 5.6.3*), the structural model of a system can be captured using static ontologies and the behavioural model can be represented using dynamic ontologies. In short, static ontologies can abstract the structural characteristics of a system. These are "the concepts of a reality and their interrelation, the static semantics of a domain, its context free or context sensitive structure" (Aβmann *et al.*, 2006:254).

Ontologies can be used to provide standardized analysis models. These include the domain-specific models that are developed by different domain experts. The addition of the specification models that translate the system requirements to domain models will yield a complete computationally independent model (CIM) (Aβmann *et al.*, 2006), as described in Section 4.3.1. This process eventually increases the number of behavioural characteristics of the software products.

### 5.7.3 Ontologies in Distributed Non-homogeneous Database Systems

Ras and Dardzinsky (2004) and Haller and Oren (2006) proposed the use of process or task ontologies in databases. Task ontologies are used to extract rules from different knowledge bases at remote sites. Although global queries can be converted to local queries, the

definitions given to these queries may vary from site to site and interpretations of these may be required. Task ontologies (Ras & Dardzinsky, 2004:55) can be used to define "the computational architecture (structure) of a knowledge system". The knowledge system performs a task in a domain defined by a set of ontologies called domain ontologies.

Before task ontologies can be used, an intermediate model, as suggested by Maluf and Wiederhold (1997), could be used. An intermediate model describes a database at very high level of abstraction, which is good enough to give a homogenous representation of all the databases in the organization.

Ras and Dardzinsky (2004) noted, however, that the problem with developing systems using task ontologies is that the ontology (domain) repository is kept in a continuous state of semantic inconsistency. Each task may generate a new definition of an already existing concept in the ontology repository. Often the repository may be updated with a somewhat global concept definition while it is, in fact, only a local definition. This problem may be exacerbated by the use of case-based reasoning (CBR) techniques that will force the designs to accept algorithm-generated semantics (Mavetera, 2007; Gomes, 2004). These semantics may not really represent the context of that domain application.

To minimize these inconsistencies, Ras and Dardzinsky (2004) suggested that the semantic changes should be tracked. At each level of information system development, the system should be able to revert to the original definitions given by the users in a specific task-related domain. Mavetera (2007) suggested that a more appropriate way would be to keep the CBR algorithm-generated ontology IS models and designs in a different database from the user-defined ontologies, as discussed in Section 5.8.

### 5.7.4 Ontologies in Systems Integration

Information systems require social and technical integration to have a fit that benefits the organization. This excerpt on system integration is also described in Mavetera (2004b). Social integration looks at the norms, meanings and group membership of people or their agents. These rules of meaning and membership could be formal or informal. Political structures, although increasingly becoming less effective and less trusted, still play an important role in social integration. Unlike social integration, technical systems integration deals with the technological means of control over the physical and social environment as well as with the skills associated with these means (Lockwood, 1964).

Since both social and technical integration are very crucial for the efficient and effective functioning of an organizational system, the facilitative power of ontologies will be discussed in this section. In system integration, ontologies can be used to allow the smooth transfer of new rules and meanings in organizational systems. On their own, ontologies facilitate the formation of alliances and the establishment of control over the resources that organizations need to achieve their outcomes. These are viewed as translations (Introna, 1997).

Most organizational systems currently in use rely on syntactic standards to work as translation and obligatory passage points. The use of syntactic standards as obligatory passage points, for example, electronic data interchange (EDI), has resulted in large enterprises being inflexible and smaller companies being locked out. In addition, the adoption of a single common standard places limits on the business models that can be supported, thus impacting on the return on investment (Smith, 2000). The use of these standards is, therefore, not strategic if different companies with disparate systems have to be integrated. The standards do not allow for the integration of different organizational political structures, norms and meanings. On the other hand, if ontologies are used correctly, they will be able to address this shortfall.

### 5.7.5 Ontologies and Software Products Reusability

In line with the philosophy of software product line approach and software kernels, discussed in Section 4.3, ontologies may be used to capture and store the analysis and design models and, in some cases, the implementation model of a system. This model base can be used as a shell for the production of other similar domain related software products. This is also stressed by Mavetera (2007) and is also reflected in Section 5.8 in the discussion of ontology-driven software architecture. This architecture has an ontology knowledge repository in which all the ontology models developed during a software development process are stored and can be reused in other software development projects. There are, of course, several other areas in which ontologies can be – and are being – used but, for the sake of this research, the study will leave consultation of the available literature to interested readers. The next section gives a brief on the role of ontologies in a romantic information system.

### 5.7.6 The role of ontologies in a Romantic Information System

As discussed in Chapter One, in agent-mediated e-market systems, the use of the schema concept, single state product spaces and negotiation environments have limited the usability of software agents in these e-markets. There is a persistent failure of e-market systems to

capture the meaning, culture, context in the brick and mortar market systems that they are supposed to represent. An ontology-driven e-marketplace will have to capture most of these requirements. For example, the product space can be replaced with ontology of products in the e-market domain. This is captured as domain and business ontologies and kept in a repository. The negotiation protocols can be presented as agent tasks, with rules and resources to use during the process. These can be captured in the system using task, process or method ontologies. For example, tasks such as negotiate discount, compare price and share deal can use resources such as product price, number of participants and delivery period. The tasks can easily be modeled as task or process ontologies that are embedded in the software agent. Intentional ontologies will model the possible life states of the negotiation environment while social ontologies are tasked with capturing the organizational culture and context of the individual organizations as well as the e-market system culture. It must be noted that all seven phases of the agent-mediated e-market framework can be automated using ontology-driven software agents and that will increase the usability of these e-market places. The whole idea is for the software agents in the system to understand each other, use tacit and intuitive knowledge to perform and execute tasks allocated to them by their human principals. These same efforts are being tried for the semantic web.

Another methodological example of ontology use in IS can be found when developing a group of related projects. In this case, ontologies allow "reuse from the modeling to implementation level" of the system development phase (Sharma & Ingle, 2011:147). It is therefore important to have a knowledge reuse development platform that can be structured in the form of the OntoSoft architecture discussed below. In theory and practice as discussed in Sharma and Ingle (2011), there are three phases where ontologies are simultaneously developed and used in software development. The first of these is the *Ontolysis* stage, a requirements engineering phase tasked with the development of domain, business and specification models of the system. This phase ensures the development of domain, business and process or task ontologies that will eventually constitute the analysis model. The second phase is termed *Ontodesign*. This process runs concurrently with the design phase, where the analysis model that is ontology-driven is formalized and all attempts are made to integrate existing ontologies. For quality purposes, ontology evaluation is also exercised. The most important issue at this stage is to pick a formal language that expressly reflects the requirements of the system under development. The third and last phase according to (Sharma & Ingle, 2011) is called *Ontocontation*. This process again runs concurrently with

the implementation phase of the system. The idea is to have an automatic generation of code and case designs as also discussed in Mavetera (2007). The good thing about this is the fact that action or process specific code is ontology driven.

The last example is the use of ontologies in software requirements engineering **(**Siegemund *et al*., 2011). Besides addressing the persistent problem of inconsistencies and incompleteness in the specification of requirements, ontologies also capture the business and domain related knowledge (Siegemund *et al*., 2011, Sharma & Ingle, 2011). This adds also to the fact that, as long as the system is in operation, requirements can never be fully specified as they will continue to evolve. Therefore, ontologies can be used for structuring the concepts, the requirements and relationships captured during the requirements phase. To illustrate this, Siegemund *et al.* (2011) developed and discussed an Ontology-Driven Requirements Engineering (ODRE) tool that, using the power of ontologies, can continuously check the consistency and completeness of the requirements gathered. This will automatically ensure the correctness of the requirements gathered. Of importance in their discussion is the idea that the requirements engineering process is a goal directed activity and this can be carried out efficiently by using business and domain ontologies during the ontolysis phase as discussed by Sharma and Ingle (2011) above. The system specification model is then added as process ontologies to completely develop the ontology-driven analysis model. These application areas are not isolated in their relevance to the need for a methodology that will be discussed in Section 6.10. Sharma and Ingle (2011) pave the way for an ontology–driven software development methodology but theirs falls short of explicitly specifying the ontologies that need to be developed at each phase and what they will address. However, they emphasize the need to capture the softer human aspects of organizational information systems into the software product. Siegemund *et al*. (2011) while supporting Sharma and Ingle (2011) went further to give a pragmatic way of ensuring completeness, hence the fidelity of the system to be developed to the real organizational information system. This can be likened to ensuring quality in the developed system. These examples are not exhaustive but assist in explaining the methodological gap that will be filled by the ontology-driven software development approach when applied in practice.

The following section will discuss an ontology-driven software development architecture that has been conceptualized to become the blue print for a software development environment.

This architecture is necessitated by the fact that most of the modern software development methodologies require an SDE.

## 5.8 Ontology-Driven Software Development Architecture

Each software development process is carried out in a software development environment (SDE). For completeness, this section will discuss a proposed SDE (Mavetera, 2007) that focuses on the role ontologies can play in software development. The architecture of this software development environment, codenamed the OntoSoft case tool (Mavetera, 2007), that positions ontologies at the centre of a software development process, is illustrated in Figure 5.3. Whitten *et al.* (2004) characterize a case tool as a software package that automates or supports the drawing and analysis of system models. In addition, case tools should provide a facility for translating system models into application programs. The OntoSoft case tool has three major components: the knowledge base repository, the designer engine and the reasoner. The knowledge base repository is discussed first.

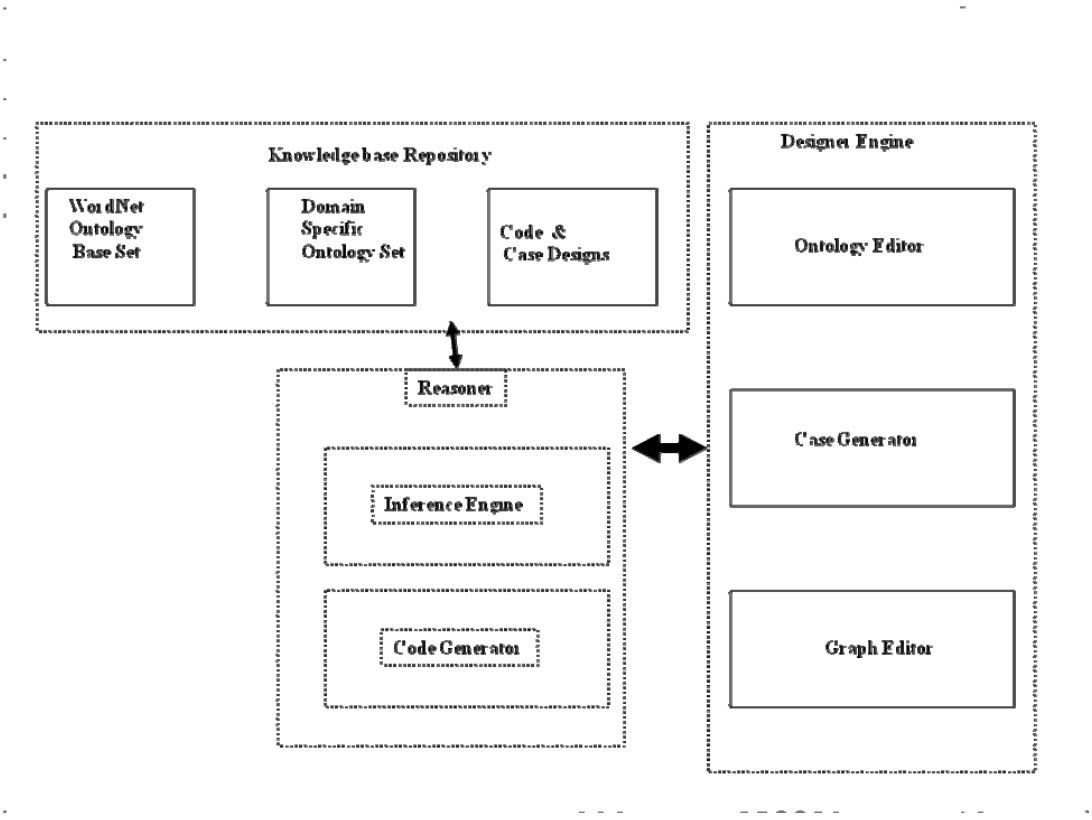


**Figure 5.3: OntoSoft Case Tool Architecture** ***(Adapted from Mavetera, 2007)***

### 5.8.1 The Knowledge Base Repository

The knowledge base repository is a facility for storing system models, their detailed descriptions and specifications. These system models are the analysis, design and implementation models explained in Section 4.3.1. The models should be captured as sets of ontologies as well. The tools and facilities for creating system models and documentation should also be stored in this repository. Unlike the Rebuilder case tool discussed by Gomes (2004), the OntoSoft knowledge base repository consists of three parts: the WordNet ontology base, the domain-specific ontology base (not found in Rebuilder or any other case tool yet developed) and a code and the case designs base set (Mavetera, 2007).

The WordNet ontology base is taken and maintained "as is" so as to satisfy the consistency concerns raised in Section 5.7.3. WordNet is a type of terminological ontology (Sowa, n.d.). It is a lexicon and consists of information about "syntax, spelling, pronunciation and usage of words". The categories in WordNet need not be fully specified by axioms and definitions. It determines the relative positions of concepts with respect to each other, using the sub-type / super-type relationships. In short, it is a natural language knowledge base. In order to maintain linguistic consistency in terms of international grammar and general meanings of terms, this ontology base should not be modified during use.

Unlike the WordNet ontology, the domain-specific ontology set is specific to an application software domain and is allowed to change according to the different conceptualizations and ontological commitments to a certain domain (Guarino, 1998). This is the knowledge base that users and developers can fine-tune to suit their application domains. Finally, the case and code designs base stores new and old designs that are relevant to a specific application domain.

### 5.8.2 The Designer Engine

This consists of an ontology editor, case generator and a graph editor. These three are used to develop domain-specific ontology case designs *'on the run'*. The design engine includes cases that capture the differing designs in the application domain, as well as graphs that are used to map related concepts in a domain through their respective conceptual relations.

The designer engine uses the same principles as Rebuilder in terms of case-indexing, the only difference being the graph editor. Rebuilder uses UML as a case editor. UML, as a case editor, does not link the cases to the meaning of the cases that are stored in the domain

ontology. It is purely syntactic. OntoSoft should use conceptual graphs that are a graphic notation for logic based on existential graphs (Sowa, 2000). The conceptual graphs are augmented with features from linguistics and the semantic networks of artificial intelligence. Conceptual graphs can be used to map to and from natural languages. As a presentation language, they are used for displaying logic in a more human readable form. The conceptual graphs are then linked to the domain-specific ontology to augment the knowledge content of the cases and designs.

### 5.8.3 The Reasoner

Lastly, the reasoner consists of the inference engine and a software code generator. The inference engine acts as a communication engine between the designer engine and the knowledge base repository. It accepts user queries, retrieves old cases and links new cases to WordNet and domain specific-ontologies and links code to the cases and conceptual graphs. In fact, it is the brains behind the OntoSoft case tool. The reasoner should be able to automatically generate software code that is related and specific to the cases that have been designed.

As such, the code generator automatically develops a code specific to a retrieved or adapted case. This reasoner uses Bayesian Networks (BN) techniques to index cases and case-based reasoning (CBR) principles which are well covered in Gomes (2004). BN uses rules of probability and are incorporated in the reasoner so that it can check for already existing ontologies, case designs, and graph designs. If there are already existing ontologies, case designs and graph designs, the reasoner should prompt the user either to accept or modify them. CBR is, therefore, based on the reuse of experience. Every instance of reasoning is an episode that is registered and stored as a case. These cases are captured as conceptual or existential graphs. It is worth noting that conceptual graphs capture different cases. The individual case captures specific situations that are context related. As each case captures a specific situation and is context related, the syntactic, semantic and pragmatic aspects of the situation are also captured (Mavetera, 2004b).

This research does not cover the concepts on CBR, BN and conceptual graphs in depth since development of ontology artefacts is not the goal. This detail falls outside the scope of the current research. The interested reader is referred to research papers by Gomes (2004) and Sowa (2000 and 2006).

It is stressed that the field of ontologies, their development and use in software development are highly dependent on the field of semiotics as discussed by Stamper (1992), Sowa (2000) and Shanks (1999). There is still some confusion as to what constitutes semiotics. The present study regards semiotics as the study of signs. These signs are used as symbols of representation in a study discipline. As the study of symbols of representation, semiotics focuses on four levels: syntax, semantics, pragmatics and social context. The discipline of semiotics is vast and an in-depth discussion of it will end up defeating the purpose of the present study. The interested reader is urged to consult the relevant literature sources and the researcher limited the scope of his research to the tenets relevant to this study.

## 5.9 The Field of Semiotics and Ontology

The field of software product development is dependent on the use and representation of the human aspect and the information technology (IT) aspect of systems using signs (Stamper, 1992). These two aspects, as arranged and described by Stamper (1992), portray the IT aspect as having the physical, empiricist and syntactic layers. The human aspect is portrayed as comprising semantics, pragmatics and the social world (context) components in a characterization called the semiotic ladder. These are also referred to as Stamper's (1992) six semiotic levels. However, somewhat thinly, Bjørner (2008) characterizes semiotics as consisting of pragmatics, semantics, and syntax only. This could be because he realized that these three are the levels most commonly used in the field of information systems and are the aspects that are probably the easiest to implement, unlike the organizational context.

According to semiotics, the study of signs (Sowa, 2000; Shanks, 1999) has three branches of representation: the first, second and third, as stipulated by Duns Scotus (1265/66-1308). The signs as symbols should be able to convey knowledge. These branches of semiotics will be discussed in the next section.

### 5.9.1 Syntactics

Syntax or syntactics refers to the representation and arrangement of signs used in a language. As Bjørner (2008:27) states, it is about representation of specifications, "rules of form, basic forms and their proper compositions". Syntax is regarded by Duns Scotus (1265/66-1308), as '*the first*', branch of representation which is called '*grammatical speculative*'. It is the language that is used to relate signs to each other and to unite material data. Syntactics is also concerned with the forms of symbols rather than with their meaning.

The focus of syntax is on the "how?" of the representation. It defines sets of symbols according to a set of formal rules. The formal definition of syntax allows symbols to be transformed from one form to another. It is important to ensure that the syntax of any language is able to be interpreted in such a way as to assign some specific meaning to the composition. The syntax allows for the compilation and processing of data in computing machines.

In information systems, formal syntax allows computers to merge, share and manipulate ontologies. As such, syntax can be attributed to the correctness and consistency in ontology coding as a form of representation. Of particular importance in syntax is the need for the ontological representation to conform to a particular grammar.

Syntax can also be regarded as signs that are used to map other signs onto a frame of reference. Sowa (2000) used the principle of '*concept of representation*' and conceptual graphs to define and represent syntax respectively.

The use of the concept of representation and of conceptual graphs goes as far as defining relationships between signs. It does not however deal with the issue of relationships among signs, the world, and the agents that observe and act upon the signs (Sowa, 2000). This symbolization uses structured similarity and indices that point at the objects they represent. Regrettably, most of the information systems in use today rely heavily on this symbolization. For example, on the Web, metadata have been used extensively to represent documents and objects using this symbolization. This type of representation does not solve the relationships between the actual objects represented by the meta-level data.

The use of metadata in information systems helps users to find information resources but these metadata are not good enough to process the information so as to find meaning from the resources. Metadata development requires participating information system developers to adhere to some representation standard. This syntactical representation is a superficial agreement. This, on its own, hides the complexities in system relationships and also reduces the interoperability of information systems, as discussed in Section 5.7.4. The next branch of representation is semantics.

### 5.9.2 Semantics

Unlike syntactics that focuses on the form of representation, semantics deals with the conditions of the truth of representations. As a branch of representation, Duns Scotus

(1265/66-1308) regarded semantics as the second issue to be considered and is defined as logic proper. In organizational systems every sign represents an object that exists in the real world. Semantics is, therefore, used to relate signs to things in the world, that is, to give meaning to symbols that represent the world view. The assignment of meaning to symbols is based on the experience and prior knowledge of the people working with the symbols (Shanks, 1999). It is also the assignment of patterns of signs to related patterns that exist among the things the signs refer to (Sowa, 2000). It is important to note that different actors sharing the same subjective world (the same knowledge and experience) may assign the same meaning to a symbol. This can be compared with the notion of common cognitive base and the definition of ontology given in Section 5.3.

In linguistics, semantics is tasked with the study and knowledge of meaning given to a language. It deals with the meaning given to signs. These signs are, however, usually represented syntactically.

### 5.9.3 Pragmatics

Pragmatics is usually referred to as the rules of effect. As discussed in Sowa (2000) and also as characterized by Duns Scotus, this can be referred to as '*the third is […] pure rhetoric'*.

This semiotic level is concerned solely with the way signs and symbols are used. In the use of symbols, there is always a generation of new signs as one works in an application area. Sowa (2000) argues that pragmatics can be regarded as a field that deals with laws that govern the birth and rebirth of new signs. The focus of pragmatics is on the effect of one sign on another sign or the cycle of one thought giving rise to another thought.

As discussed by Bjørner (2008), pragmatics can be viewed as the study and practice of factors that direct our selection of language in social dealings and the effects of our choice on others. This indicates the use of a language in a social context, that is, decisions made to the choice of specific expressions, motives behind specific utterances and expressions. As Bjørner (2008:27) puts it, pragmatics concerns itself with "bridging the explanatory gap between sentence meaning and speaker's meaning". Pragmatics can also be considered as the rules of effect, of influencing others by what we say and how we say it. Sowa (2000) summed it up by suggesting that grammar should be the code for pragmatics.

When applied to information systems environments, pragmatics brings forth the contextual issues that affect a symbol in and around its environment. At this level, the representation

should have a purpose and be useful and usable. Most importantly, the suitability of the representation to the task at hand is brought to the fore.

### 5.9.4 Social Aspect (Context)

The social aspect is concerned with the context in which a sign or symbol is used. It also deals with the understanding of the meaning that people get when they use symbols. As in semantics, two actors can share the same subjective world, but if they are to assign similar meaning to the symbols, their viewpoints, biases, cultural and political issues have to be the same. A shared understanding of '*the represented*' should be achieved. In other words, the social semiotic level is concerned with the shared understanding in the organizational business world. It is, therefore, important that the cultural and social context of situation be taken into consideration.

This discussion on semiotics is an attempt to find the necessary grounding that can be used to characterize ontologies from a structural linguistic viewpoint. It should be noted that many ontology definitions use a lot of mathematical formalizations, thereby limiting their understandability to practitioners in the hard-design sciences, such as computer science, artificial intelligence, description logics, to name but a few. A softer characterization, based on the field of semiotics is required if ontologies are to be used in methodological fields such as software development approaches. The discussion on the structural aspects of ontologies will be used to find a working definition of ontologies as used in this study.

## 5.10 Towards a Working Definition of Ontology

The problem with the development of information systems has been the lack of a linguistic model, either informal or formal, that has sufficient translations to allow the exact mapping of the real world view to the system view. In the design of spatial information systems, Molenaar (1998:195) proposed a linguistic model with three components, syntax, semantics and uncertainty, as shown in Figure 5.4. In his discussion Molenaar (1998) says that syntax describes how to link object identifiers to information about the object classes, thematic attributes and geometric data for the formulation of statements about those objects and their mutual relationships.

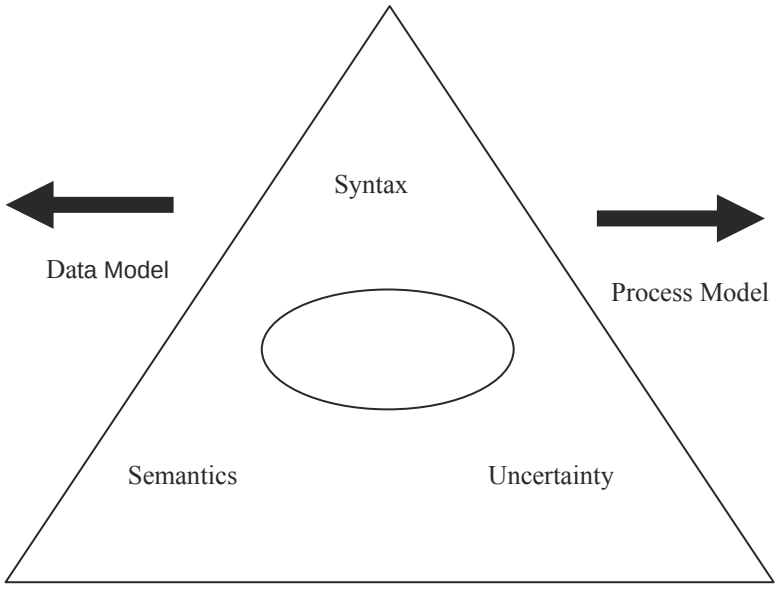


**Figure 5.4: A Linguistic Model (*Adapted from Molenaar, 1998*)**

He also portrayed semantics as the link between database representation and a real world situation. In addition, he also noted that the use of data models requires real world phenomena to be described in discrete categories. However this description will always have a certain degree of indeterminacy. Data models, therefore, do not represent reality with absolute precision or certainty. Molenaar (1998) also affirms that all data in information systems refer to objects that are conceptual entities within a certain semantic framework defined within some user's context.

Using context as the focus point, conceptual entities defined by users are, therefore, not very specific. The result is that, in the statement about real world phenomena, terms portrayed in information systems are formulated and understood within a certain application context (Molenaar 1998). This argument shows that any semantic description to be handled by the syntax should also include some uncertainty (fuzziness) inherent in the description of reality as models.

Molenaar (1998) also contends that any linguistic model should be able to handle the correlations of the three components: syntax, semantics and uncertainty, as illustrated in Figure 5.4. Any implementation model for information systems should also be able to incorporate both data and process models within this linguistic model. It is quite uncommon

for Molenaar, as a researcher in the field of natural sciences, not to mention the role of pragmatics in spatial information systems.

In another discussion, Mavetera (2004b) conceptualized a romantic information system development (ISD) model that argued for a linguistic model consisting of syntax, semantics, pragmatics and uncertainty. This linguistic model, without the uncertainty, represented the ontology model. Mavetera (2004b) further argued that, from an ISD paradigm, strategy and methodological perspective, different developers have been addressing the data, process and context needs of systems separately. This has limited the usefulness of the resultant information systems and their software products to a great extent. Furthermore, Mavetera (2004b) did not mention the role of social context in information system in addition to its role on the sound definition of the ontology model presented.

Aβmann *et al.* (2006:256), however, characterize an ontology as a linguistic model, as a "shared, descriptive, structural model, representing reality by a set of concepts, their interrelations, and constraints under the open-world assumption". As discussed previously, the open world assumption purports that anything that is not explicitly expressed by the ontology is, therefore, (considered to be) unknown. Any aspect of the world view that is known should be captured and documented expressly and explicitly. The intensional conceptualization of ontology models all the features, the known and the unknown, while the extensional conceptualization only captures the known (see section 5.5 above).

This definition argues that the ontology has to be shareable and possess the behavioural characteristics of the domain. Considering the discussions in Section 5.6 (ontology varieties), Section 5.7 (possible ontology uses in software development), Section 5.9 (semiotics) and the work on this section by Mavetera (2004b), Molenaar (1998) and Aβmann *et al.* (2006), the following persistent sticking points are noted about software products, systems and the ontology model:

- Contextual representation requires the capturing of syntactics, semantics, pragmatics and the social context of an organizational system.

- Owing to the existence of informal communication in organizations and some degree of indeterminacy in eliciting implied meanings, there is also some degree of uncertainty or fuzziness in any model that can be used to represent a system in the real world.

- A representation or model whose characteristics are grounded in the relativistic world view that falls in the open-world assumption is required to characterize the real world from the interpretive or neo-humanistic paradigm stance.

- A linguistic model, although being an intensional definition and not an extensional definition of the real world, is required that allows a shared, descriptive, structural and behavioural model of the real world to be captured.

In short, this linguistic model is called ontology and is characterized in softer terms as follows:

> *Ontology is a linguistic model of the world that comprises of syntactics, semantics, pragmatics as well as the social context of that which is represented. Despite some unavoidable informal indeterminacy in the real world view, it should allow a shared, descriptive, both structural and behavioural modelling and representation of a real world view by a set of concepts, their relationships and constraints under the open world assumption.*

This characterization is very important to people who work in the softer fields of IT such as information systems. It is a characterization that allows behavioural and constructivist scientists to develop frameworks that can guide the subsequent development of IT artefacts by the design scientists. Furthermore it also allows them to compare real-life occurrences with what ontologies stand for. For example, in the field of software development, ontologies as a knowledge representation model, communication model, development technique or tool can be contrasted to most of the challenges that developers meet during their daily practices. This definition will be put to use in Chapter 6, in which an ontological approach to software development is formulated.

## 5.11 Summary

This chapter outlined several definitions that are given to the term ‘ontology’. These range from the fields of philosophy to computer science and information systems. It is argued that, although the philosophical ontology is the basis upon which all the other applications of ontology are grounded in; there are still some differences, depending on the field of application. For example, the purely technical definition of ontology in computer science and artificial intelligence fields may find very limited application in softer fields such as software

and information systems development. A more informal definition, as proposed here, may be more appropriate. This discussion on ontology, besides adding theoretical sensitivity to the researcher, was used as a secondary data input data input during the conceptual mapping process, as will be described in Chapter 6.