Report # 2

(June 4th - June 8th, 2007)

Raghav Krishna Padmanabhan

TANGO,

Doc Lab,

Rensselaer Polytechnic Institute.

This report details the study I have done in the past week (June 4 - 8 ,2007). It is divided into two parts. The first one deals with Object-oriented System Analysis. The second one deals with my study and understanding of Ontologies, so far.

PART 1:

Object-oriented System Analysis (OSA)

The word 'System' refers to a group of objects that interact with each other. System Analysis is the study of a group of interacting objects in a specific domain to understand and document their essential characteristics. It is important that one understands the software system completely before he/she proceeds to design it. There are different methods of system analysis like Natural Language Processing, Process Oriented Analysis and Object oriented Systems Analysis.

Why OSA?

Natural Language Processing is the method of describing the system in natural language. The description for large systems ran through pages and the functionality of the system was spread throughout the document. This makes it difficult to map the concepts to the real world objects.

The Process Oriented Analysis broke down the system into a network of interacting processes. The disadvantages with this method are that
(a) The process interact with multiple real world objects making the process of understanding the system fuzzy.
(b) This method makes the user focus on design pre-maturely. If there are a lot of processes involved, the analyst tends to look into *how* the processing should be done and other details. In Systems analysis, our primary aim should be *what* the system is and how the objects interact.

The Object-oriented System Analysis or OSA separates the document into distinct,self contained elements along the same boundaries that exist in the real world. Thus the entire information regarding the system object can be found without any hassle at a single logical location which simplifies conceptual correspondence. A *reality model,* OSA helps the analyst to focus on the *what* rather than the *how.*

The components of OSA:

OSA can be divided into three components :

1. Object Relationship Model.

2. Object Behavior Model.

3. Object Interaction Model.

I studied the three component models briefly and their overview is presented below.

Object Relationship Model:

The foundation for OSA is the object class. Object class represents a group of system objects with similar characteristics and behavior. The members of the Object Class are called the Instances of the Class. System Objects rarely stand alone. They exhibit relationships with the other system objects. The group of relationships is called a relationship set. An Object relationship model describes the relationship between a group of system models. I found parallels between this model and the Entity Relationship Diagram in Database Systems.

The Object Relationship model is described using the ORM diagram.

<u>Object-Behavior Model:</u>

These models describe the Behavior of the objects in the system. The intensional meaning of Behavior in the context of System Analysis is explained below.

What is Behavior?

(a) The various states that the system object exhibits throughout its existence. (keyword: States)

(b) The conditions that cause the system object to change its states. (keyword: Transient Conditions)

(c) The actions performed on/by on the object . (keyword: Actions)

We can document States, Transient Conditions and Actions in models called "State Nets".

I had a little difficulty in differentiating between Transient Conditions and Actions which I think will be clear on reading more on OBM examples.

Large Software systems require some additional concepts like Views and High Level concepts to be implemented to represent them. These concepts allow the analysts to use the feature of abstraction and enable him/her to analyze the most salient features of the model. However, if it is a complex model and the details of some of the high level objects are required, the analyst may expose parts of it for better understanding.

<u>Object Interaction Model:</u>

Objects interact with other objects.This interaction is modeled by the Object Interaction Model (OIM). This is done using the interaction diagrams. The interaction links are represented using zigzag lines. In case in the interaction diagram, the analyst wants to know more about the object class Behavior, he/she can do that by looking at the State Net of the object.

The following section describes my understanding of the Object Relationship Model.

An Object is a single entity or notion. It can take physical or conceptual manifestation or could be an event. Examples include a car, John's weight or the take off of a flight. In natural language analogy these are similar to *"Nouns"*. Objects are represented through dots. The name of the object is specified below/adjacent to the dot. Since a standalone object on its own cannot be of much use. We need to have objects interacting with each other. A relationship establishes a logical connection between objects. In natural language analogy these are akin to *"Verbs".* Relationships are represented using lines and the name is specified on the line.Examples include (Raghav) 'owns a' (BMW), (RPI) 'has a' (Gym Facility). The number of connections to objects is called the Arity of the relationship. If there are two connections, such a relationship is a binary relationship. If an object is connected to two objects, it is a part of a ternary relationship.

When it comes to analyzing large systems, a level of abstraction is necessary. This would be grouping a large body of facts into smaller comprehensible units i.e. A set of objects are grouped together to for a logical reason and called Object Sets. Object Sets are represented using rectangles with the name inside it. The name of the object class should be specific,well qualified and suitable to be applied to a specific object. (Example: It is advisable to name the object class as "Car" rather than "Cars"). The abstraction when applied to Relationships, we obtain Relationship sets. Relationship Sets are represented using diamond connecting the object classes.(There is a degree of latitude allowed when it comes to representing them for simple binary relationships - omit the diamond and connect them using lines).

To explain models better, the OSA has constraints that are applied on Object and Relationship Sets. Constraints are imposed on Object and Relationship Sets and restrict their membership. There are 3 types of Constraints:

(a) Participation Constraints:

Participation Constraints defines the number of times an object in an object class can participate in a relationship set. It is represented using the notation "*min:max". Min* represents the minimum number of times the object can participate and *max* represents the maximum number of times the object can participate in the relationship set.

(b) Concurrence Constraints:

Concurrence Constraints restrict the number of times an object can appear along with other objects in a relationship set. This constraint specifies the number of times a particular group of objects can appear in a relationship set.

(c) Object Class Cardinality Constraints:

Object Class Cardinality Constraints specify the number of objects that can exist in an Object Class .

Special Relationship Sets:

OSA recognizes certain special relationship sets which commonly occur while modeling systems and places special representations for them. They are:

(a) Generalization/Specialization

(b) Association

(c) Aggregation

1) Generalization Relationship Set :

This relationship set represents a " IS A " relationship set. In this relationship set, one object class is categorized as a "*subclass*" of another class. The "*superset*" object class is called the Generalization class and the "*subset*" object class is called the Specialization class. This relationship set is denoted by a transparent triangle with the top vertex pointing towards the Generalization object class. A generalization object class may contain more than one specialization class. One important observation about the Specialization object class is that the participation constraint for it is always 1.

There are specific constraints associated with the Specialization Object Class:

(a) Union Constraint

(b) Mutual Exclusion Constraint and

(C) Partition Constraint

Union Constraint :

A Union Constraint when imposed on a specialization tells us that every member of the generalization should be a member of one of the Specializations. In other words, the union of the group of specializations constitutes the entire membership of the specialization. This constraint does not preclude the scenario of one specialization class object being a member of another specialization. Indicated with a "U" inside the triangle.

Mutual Exclusion Constraint :

A Mutual Exclusion when imposed on a group of specializations tells us that any member of a specialization object class cannot be a member of another specialization class. In other words, the Mutual Exclusion Constraint declares that a group of specializations of a generalization are pairwise disjoint. Indicated with a "+" within the transparent triangle.

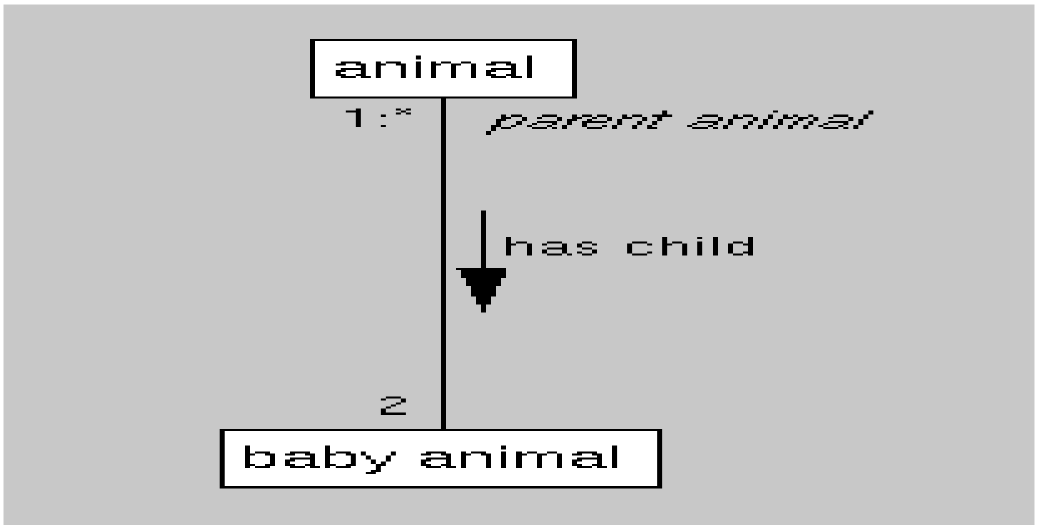
Partition Constraint:

A partition constraint declares that a group of specializations partition a generalization. A partition constraint is said to be imposed on a group of specializations if they have a Union Constraint and a Mutual Exclusion constraint associated with them.

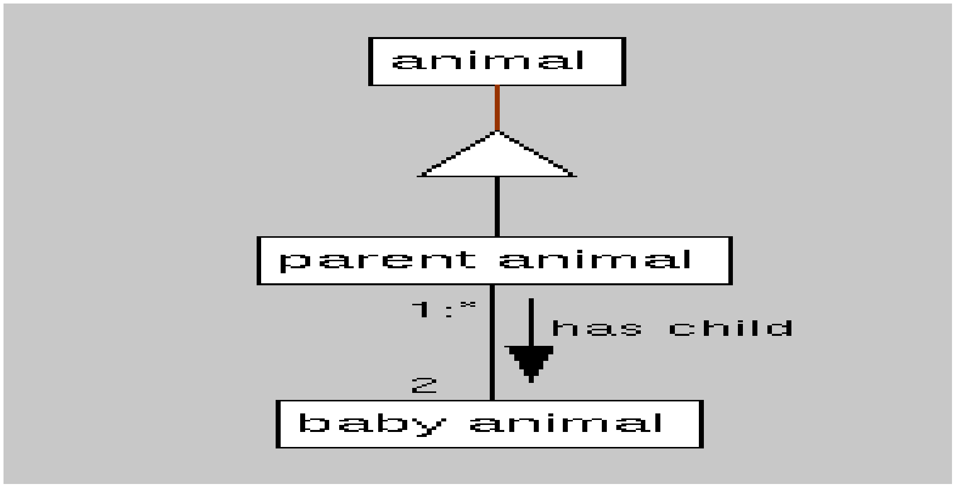
Roles:

A role is a label that is placed on a relationship set that gives a special name to all objects participating in the relationship
A Role is a specialization object class of a generalization whose members fully participate in the relationship set in the connection.

Consider a generalization "Animal" in a relationship set - "has child" - with another Object Class "Baby Animal". Now, every member in the animal object class need not participate in the relationship. We define a role "Parent Animal" such that every parent animal definitely participates in the relationship set.



A role is really just a shorthand way for modeling a generalization/specialization relationship. The original example could have also been modeled as follows:



Inheritance:

Inheritance is the property based on which each specialization object participates in all the relationship sets of its generalization.

2) Aggregation Relationship Set:

This relationship set represents "IS PART OF" relationship. It is represented by a Solid Filled Rectangle. We use this relationship set to denote that a particular Object Class is composed of various other (sub)Object classes. IN other words, The Superpart is composed of Subparts. The Super part is called the Aggregate and the sub parts are called the Components.

The participation constraints of the Aggregation set are represented just below the filled triangle. It should be remembered that the collection of the subparts need not constitute the superpart.These subparts may only be the sub parts we choose to represent in the model.

3) Association Relationship Set:

This relationship set represents "IS A MEMBER OF" relationship. This is used when we want to form a set of objects that we wish to consider as a single object.
In the two object classes, the objects of one object class is a set whose members are the objects of the other connecting object class.

The object class whose objects are sets is called the Set Class / Association while the object class whose objects are members of the set is called the Member Class / Universe.
Association is a subset of Universe. Association is represented by a line with a * at the set class end.

This concludes the Part 1 of the report. I plan to study about the Object Behavior Models and Object Interaction Models soon. The Part 2 of the report discusses Ontologies and my understanding of them.

PART 2 :

ONTOLOGIES

I read the discussion of Ontologies by Yuji Tijerino to understand more about Ontologies. The following report describes what i have understood about them.

What is an Ontology?

A seemingly simple question with a lot of possible answers and each of them subject to some level of criticism.

In philosophy it means theory of existence, a definition which I did not further delve into at this point of time.(But I decided to come back to this probably when I begin to use walking stick for support.)

Other definitions relevant to the context of Computer Science:

(a) An Explicit Specification of a Conceptualization - a widely accepted definition in the AI community.

(b) A theory or a system of concepts/Vocabulary used as a building blocks of an information system. In addition this definition also entails two kinds of ontologies in the context of problem solving. Task Ontology for problem solving processes and Domain Ontology for the domain where the task is performed.

(c) Agreements about shared conceptualizations.

(d) A hierarchical organization of concepts, relations among the concepts and axioms to formalize relations and definitions.

Of these, I was not able to understand definition (c).

Before proceeding to read further about ontologies, I realized that there were certain fundamental definitions that I needed to know and understand. A few of them are given below:

(1) Intension :

Intension is any property,quality connoted by a word, phrase or any other symbol.

(2)Extension :

Extension is the set of all actual things that the word or phrase describes.

(3) Conceptualization :

An abstract simplified view of the world that we wish to represent for some purpose.

(4) Intensional Semantics :

The framework that models the meaning of sentences based on the intensions of linguistic expressions.

(5)Extensional Semantics :

Framework that models the meaning of sentences based on the extensions of linguistic expressions.

In our context we are concerned with ontologies for data extraction and hence refer to them as Data Extraction Ontologies. For extracting information, data extraction ontologies emphasize on the role of Intensional and Extensional Semantics. An Extraction Ontology has Object sets or Concepts (analogous to Object Class in OSA) and Relationship sets (similar to the Relationship Sets in OSA). An Ontology has two kinds of concepts: Lexical and Non Lexical concepts . Lexical concepts are those whose instances are indistinguishable from their representations. Non-lexical concepts are those instances represent real world objects. (I browsed through several papers in linguistics to obtain a formal definition of lexical and non lexical concepts to formalize the language to be used in the discussion but was not entirely successful and hence the further discussion here will be based on my understanding of them.) Non lexical concepts are of paramount importance to the data extraction ontologies as they bridge the gap between the intensional semantics of the ontology and the target objects present in the target data extraction source.

It is also very important to understand what an Ontology is not in addition to understanding what it is to avoid any misconceptions.

1. An ontology is not just a set of terms or words. It is a Theory of Concepts. The words or terms in the Ontology are just to make it '*People Readable'.* The formal meaning is to be understood by the machines for the Ontology to be effective.

2. An ontology is not just a hierarchy of concepts. Although all ontologies include an IS A taxonomy, all IS A taxonomies cannot be considered as ontologies.

3. An Ontology is not just a set of axioms although axioms may be included in ontologies to give formal definitions for certain concepts involved in the ontology.

4. A Knowledge Representation (like OWL or RDF) should not be confused with an ontology. Whether a certain thing is an ontology or not is independent of how it is represented.

5. An ontology is not conceptual schema for a database. Databases are centralized information repositories under a rigid conceptual schema and whose information is not likely to be shared by other databases. Even if they are shared, the sharing of the concepts are not explicit specifications.

6. A main characteristic of an Ontology is its ability to share the information it contains across other

ontologies. Thus, by definition an ontology should be '*Open'.* And if an ontology stands by itself, if it is *closed*, it is not an Ontology at all .

I realized that there is still a lot of work left in understanding ontologies and how they work. I hope to interact with Dr. Embley closely during his visit here to RPI and get a clearer picture.

In this week I also started working with several test tables for the Wang Notation Tool and identified a few areas of concern and a few other features that would make the tool even more effective. I shall probably present them in further reports when I have done considerable amount of testing using different tables.