A COMPARATIVE ANALYSIS TO VALIDATE THE BENIFITS OF FORMAL VERSUS INFORMAL SOFTWARE MODEL TRANSFORMATION

Kaden Daley and Emanuel S. Grant
University of North Dakota
Department of Computer Science
Grand Forks, North Dakota, USA
kaden.daley@und.edu grante@cs.und.edu

Abstract

In object –oriented development the Unified Modeling Language (UML) is the ISO/IEC standard for modeling language. In relational database development, entity-relationship models have traditionally been use for modeling such systems. Transforming from one notation to another notation is of great importance, prior assessment has been carried out on transforming UML class diagram models to object-oriented relational databases, which yield significant results.

A prior research effort examined the benefits of two such approaches in the transforming process, sought to determine the benefits of an approach, by way of comparative analysis. The researchers drew inferences from the comparative analysis as to the suitability of one approach versus the other on classes of problem domains. There will be an attempt to apply a similar comparative analysis on a model from a different application domain. The goal of this research is to provide further validation of the usefulness of this type of comparative analysis.

1 Introduction

The evolve of object-oriented (OO) modeling, the Object Management Group's (OMG) [3] Unified Modeling Language (UML) [8] has evolved into the *ISO/IEC* modeling language for such systems. Entity- Relationship (E-R) and Extended-ER (EER) are relational database modeling notations but of these two, ER modeling is the more widely used designing method for relational databases. EER modeling is the enhanced ER modeling that supports class representation [4]. With the acknowledgement that relational database management systems need a better representation of the real world than that are derived from the E-R models [2]. Object-oriented modeling is an effective technology for representing real world structures as it specifies multiple views of a system through a series of models. There are a number of object-relational databases such as IBM – DB2, Oracle, with Oracle being the most popular in the market at present. Though object-relational databases use object-oriented concepts, they still apply a tabular representation.

Considering EER, ER, and UML, graphical notations for relational databases textual specifications can be directly used to create the tables. However, graphical representations provide greater scope for a better design of a system.UML is the most widely used notation for modeling classes and relationships between classes, also UML support association class while showing multiple relationship [1], and of such will be the modeling language for this paper. Hence, when UML is used to model database systems, there is a need to transform the UML models to object-relational database representation.

In this work, we are proposing to compare and evaluate techniques for transforming UML class diagram models into object-relational databases. This work is a comparative analysis of formal and informal techniques for transforming UML class diagrams into a set of SQL:1999 [9] statements. With the completion of an earlier paper by R. Chennamaneni [7], using both the formal and the informal technique where an airline reservation system was used as the bases of this comparative analysis. This paper will outline a similar technique but with the use of another system, Unmanned Aerial System, where data from this system will be used to confirm the work that was done earlier. The focus will be similar to comparing the resulting SQL statements using one of the transformation techniques.

This paper is organized in the following format: Background, which will provide concepts related to this research paper. The following section will look at the Informal Transformation Technique details, followed by a conclusion.

2 Background

2.1 Unified Modeling Language

UML- Unified Modeling Language, the standard language used for modeling business and software application needs[1], this view is also beckoned as the way the world models not only application structure, behavior, and architecture, but also business process ,data structure and data schemas[2]. UML is build on a four layer meta-model structure, which is demonstrate in figure 1 below, where M0, M1, M2, M3 refers to user objects, user-defined model, meta- model, and meta-meta-model respectively [3].

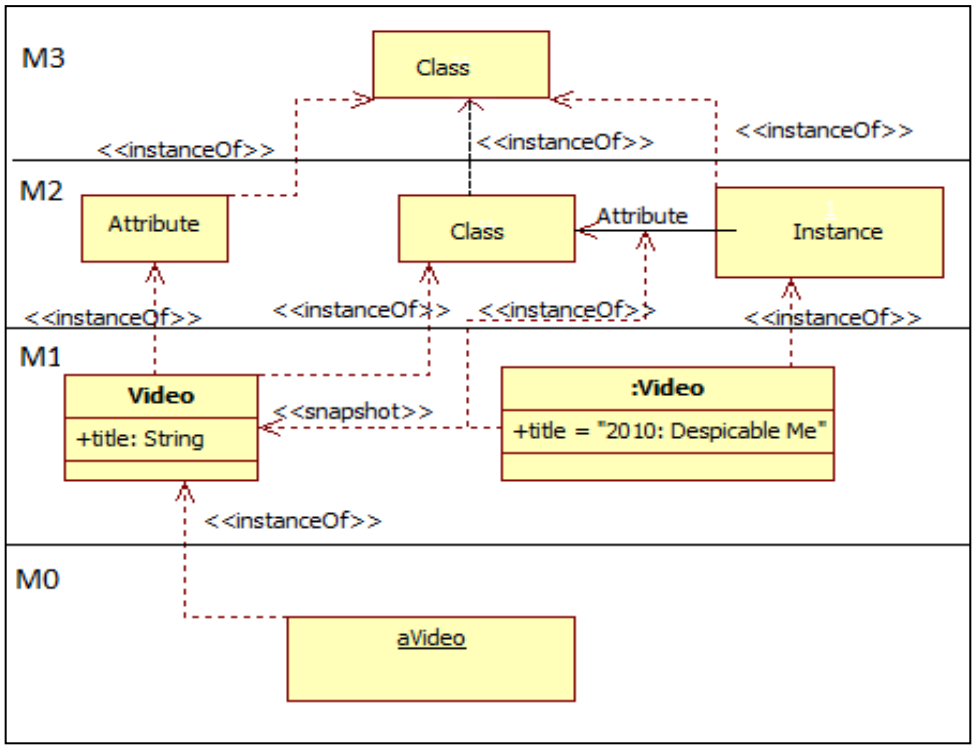

Figure 1: Four- layer metamodel hierarchy

The current version of UML is UML 2.3 which was released in May of 2010. This version consists of thirteen basic diagrams, which are divided into two sections, namely structural modeling diagrams and behavioral modeling diagrams. Within structural modeling the focus is on defining the static view of a system, which is suitable to model relationships between elements. Types of structural models includes: Package diagrams, class diagrams, object diagrams and component diagrams. The essence of behavioral diagrams is to look at the action, through observation through interaction within the real – world or operations over a period of time. Use case diagrams, activity diagrams, sequence

diagrams, timing diagrams are some of the diagrams which belong to Behavioral Modeling group [2].

Based on the various diagrams mention above, the researchers aim is to model a static view; therefore structural model is well suited. Class diagrams display the properties for model interfaces and their relationships therefore this paper will focus on the use of this diagram throughout this paper.

2.2 Class Diagrams:

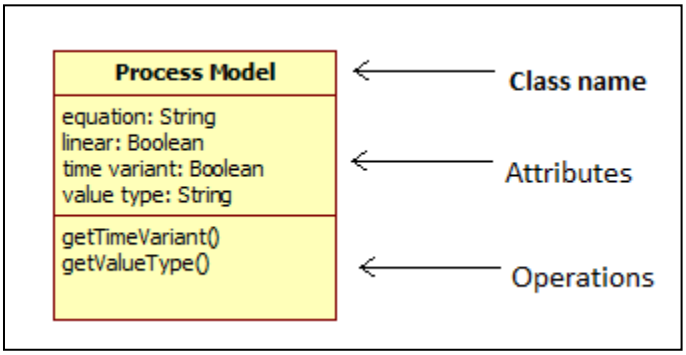

Figure 2: Details within a Class Diagram

A class diagram provides a static view of a model, describing the attributes and behavior of a model. Class diagram graphical properties allow the user to illustrate relationships between classes and interfaces. Figure 2 above shows an example of details found within a class diagram. The use of constrains, tagged values and stereotypes may be used to define classes.

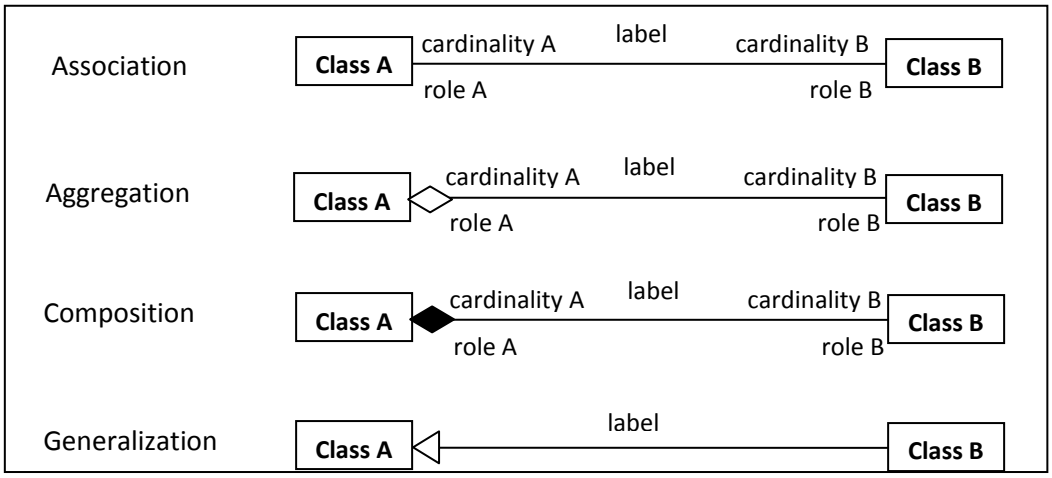

Figure 3: Notations of relation

Figure 3 shows some classification of Relationships, where association shows relationship between two or more elements. Aggregation is considered as a special case of association. Composition is a strong form of aggregation and Generalization shows relationship between a super and a sub-class [1]. Table 1 shows the types of cardinalities along with their meanings that may be imposed on a relationship.

0..*	Zero or more
0..1	Zero or one
1..*	One or more
1	One only
n	n only (n > 1)
0..n	Zero to n (n > 1)
1..n	One to n (n > 1)

Table 1: Types of Cardinalities

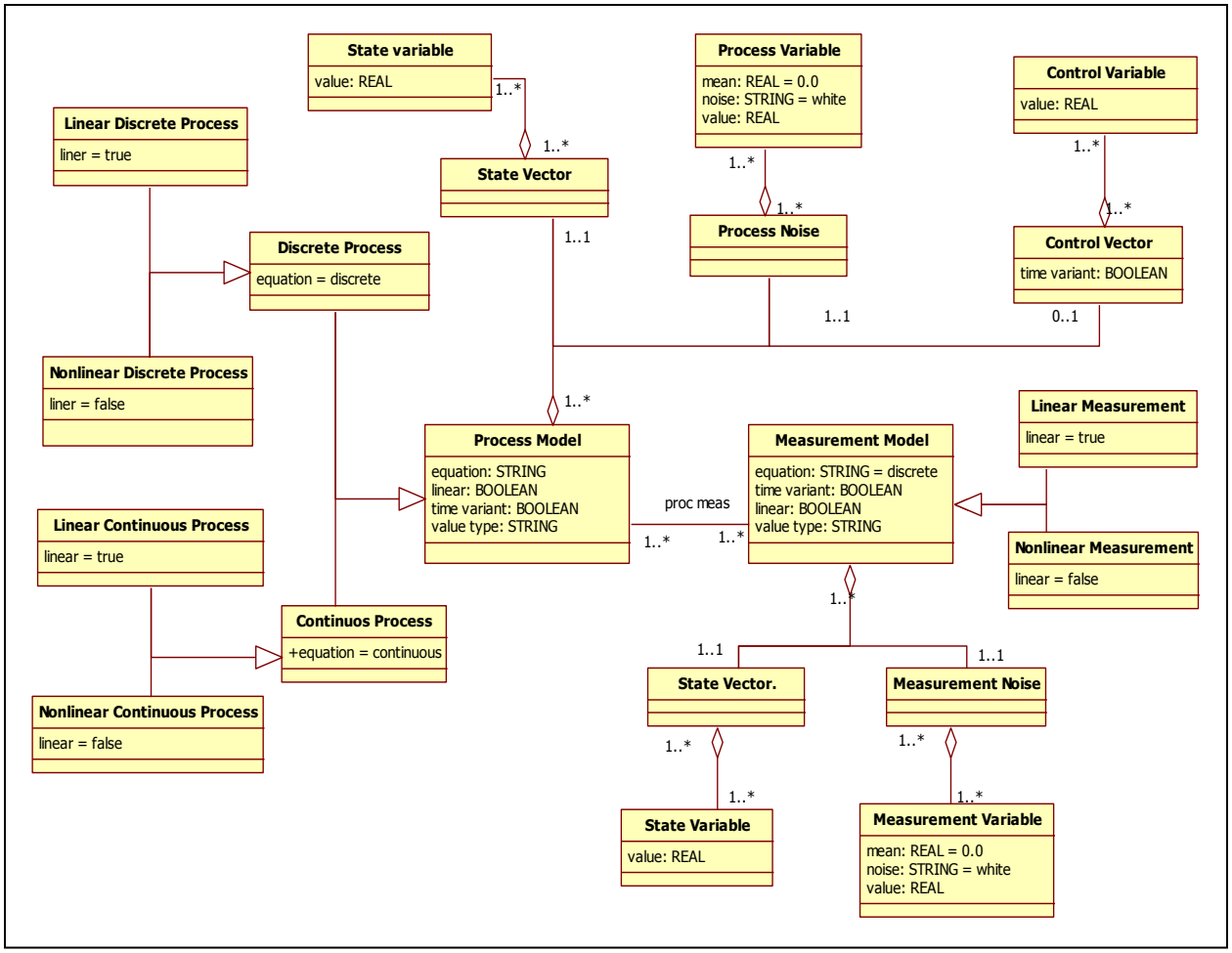

Figure 4: Class diagram of Unmanned Aerial System Operator Display

Figure 4 shows an Unmanned Aerial System Class Diagram which will be used for analyzing and evaluating the methodologies.

2.3 Normal Forms

The process of normalization can be viewed as examine relation to reduce redundancy and provide a more efficient database arrangement [4]. Table 2 below shows examples of some normal forms.

Form	**Meaning**
First (1NF)	Relation contains no non-atomic attributes or nested relations
Second(2NF)	Relations where primary key contains multiple attributes
Third (3NF)	Relation should not transitively dependency on a non-key attribute on the primary key

Table 2: Normal Forms

3 Informal Technique (ITT)

The ITT is used to transform UML diagram to object relational schema by utilizing UML extension mechanisms (stereotypes, tagged values and constraints) to define a new set of UML model element that represent object-relational database concepts of SQL:1999. These new model elements are used to derive object-relational databases. [5]

SQL:1999	**UML element**	**Stereotypes**
Database	Component	<<Database>>
Schema	Package	<<Schema>>
Tablespace	Component	<<Tablespace>>
Table	Class	<<Table>>
View	Class	<<View>>
Index	Class	<<Index>>
Column	Attributes	<<Column>>
Primary Key	Attributes	<<PK>>
Foreign Key	Attributes	<<FK>>
Multivalued Attribute	Attribute	<<AM>>
Calculated Attribute	Attribute	<<AD>>
Composed Attribute	Attribute	<<AC>>
NOT NULL Restriction	Attributes	<<NOT NULL>>
Unique Restriction	Attributes	<<Unique>>
Trigger	Restriction	<<Trigger>>
Restriction	Restriction	<<Check>>
Stored Procedure	Class	<<Stored Procedure>>

Table 3: ITT Stereotypes for database design

The extensions listed in Table 3 above, can be used to design object-relational databases. Additional extensions for collection types, methods, attributes for object-relational database is listed in Table 4. The guidelines for object relationship are provided in Table 5.

Using the class diagram from figure 4 with the extension mechanisms, yielded Figure 5.

SQL:1999	**UML element**	**Stereotypes**
Structured Type	Class	<<udt>>
Typed Table	Class	<<persistent>>
Composes	Association	<<composes>>
REF Type	Attribute	<<ref>>
Array	Attribute	<<array>>
Row Type	Attribute	<<row>>
Redefined Method	Method	<<redef>>
Deferred Method	Method	<<def>>

Figure 4: ITT Stereotypes for object-relational databases

UML	**SQL:1999**
Class - Class Extension	Structured Table Typed Table
Attribute - Multivalued - Composed - Calculated	Attribute -Array - ROW/Structured Type in column - Trigger/Method
Association - One-to-One -One-to-Many - Many-to-Many	 REF/REF REF/ARRAY ARRAY/ARRAY
Aggregation	Array
Generalization	Types/Typed Tables

Table 1: ITT Guidelines for object-relational databases

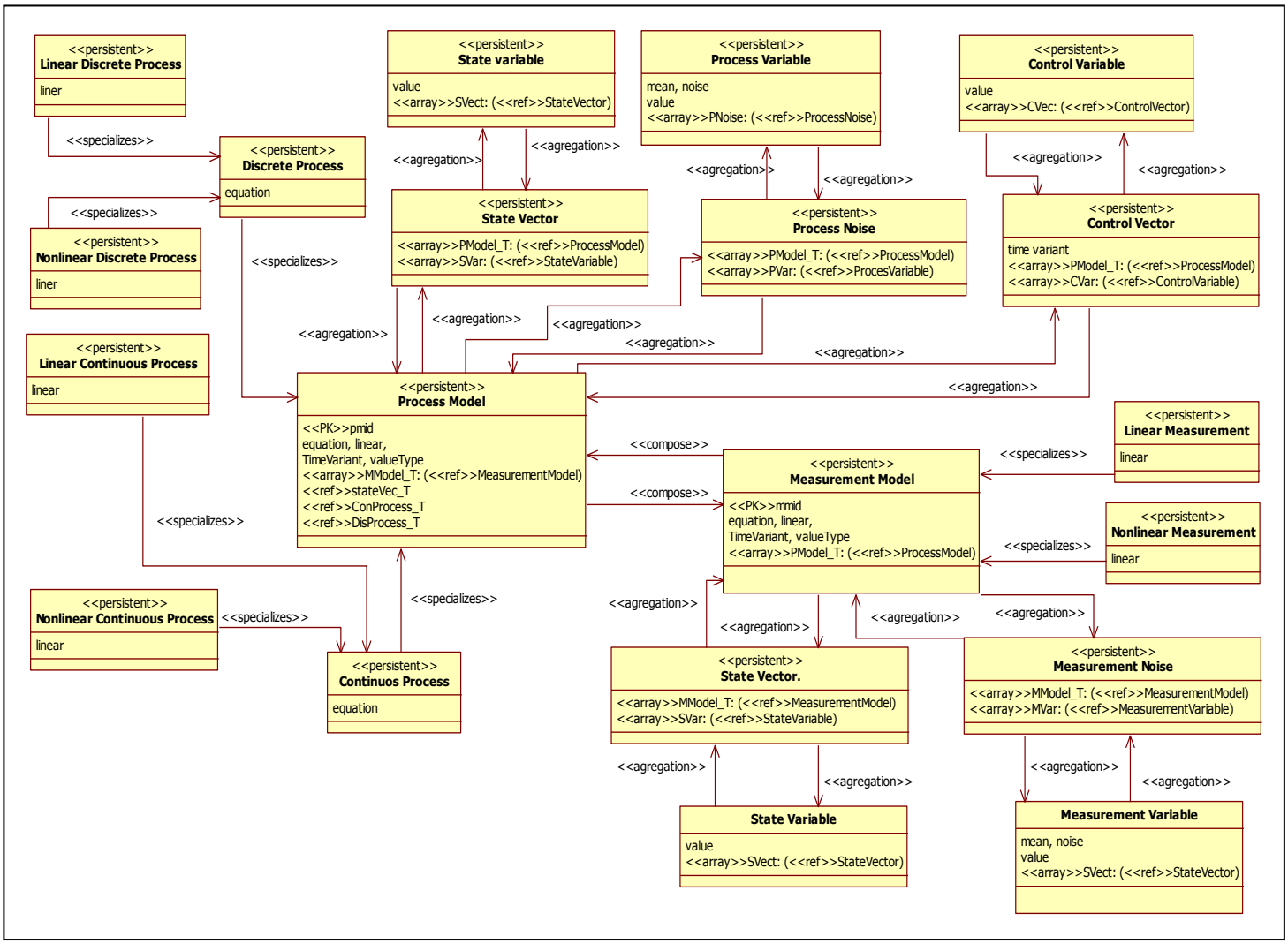

Figure 5: UML design using ITT

4 Conclusions

The use of the informal technique showed promising results, in that the transformation from the class diagrams to a UML design was able to be accomplished at this time. The objectives of this paper was not accomplished due to the time taken to designand convert the class diagrams into UML design using ITT. Comparing Figure 5 to the UML design for Airline Reservations using Methodology 1 of R. Chennamaneni [7] work, there are some differences with formation of the diagrams. Future work will have to be carried to clarify these conflicts before future designs may be accomplish.

The implementation of Figure 5 using SQL: 1999 will be developed in future work. Also, similar transformation technique using the formal technique will be explored. Accomplishing this, the researchers will be able to analyze the domains from work done by R. Chennamaneni [7] with the Unmanned Aerial System.
A formal method to transforming class diagrams to SQL statements will be examined.

Acknowledgement

Significant help was provided by the Intercollegiate to offset some of the cost associated with presenting this work.

References

1. E. Naiburg and R. Maksimchuk, "*UML for Database Design*", Addison-Wesley, 2001
2. www.sparxsystems.com
3. www.omg.org, Object Management Group
4. R. Emasri and I. Navathe, "*Fundamental of Database Systems fourth Edition*", Addison-Wesley, 2004
5. E. Marcos, B. Vela and J. Cavero ,"*Extending UML for Object-Relational Database Design"*, 2000.
6. P. Chen, "*The Entity-Relationship Model-Toward A Unified View Of Data*", *ACM Transactions on Database Systems,* 1(1) 1976, 9-36
7. R. Chennamaneni and E. Grant," *Comparison and Evaluation of Methodologies for Transforming UML Models to Object-Relational Databases*", 2001.
8. www.uml.org, Uniform Modeling Language
9. A. Eisenberg, and J. Melton (1999). *SQL:1999, formerly known as SQL3*.