EVALUATION ON THE CATEGORICAL DBMS FOR THE VIRTUALGPS KNOWLEDGE SYSTEM

Y.Xu, Z.Xu, X.Jiang, P.J. Scott and Pickering J.
University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK

ABSTRACT

The Geometrical Product Specification (GPS) is a set of standards widely adopted by modern manufacturing industries for expressing tolerances and communicating geometrical requirements. It has great benefit in reduction of manufacturing cost, alleviation of design and manufacturing misunderstandings and insurance of the consistency of geometric characteristics for the machining products through the manufacturing life-cycle. The development of a virtualGPS knowledge based system to facilitate the usage of the GPS and the training of new engineers is described in this paper. This paper focuses on giving a short evaluation of the so-called "categorical" Database Management System (DBMS) developed for storage and retrieval of data for the system.

Keywords GPS, Category Theory, DBMS, Categorical

1 Introduction

In modern industries, all manufacturers are applying various tools and methods to ensure the consistency of geometric characteristics for the machining products through the manufacturing life-cycle. To ensure the consistency of geometric characteristics, a universal accepted language should be adopted to precisely transform functional requirements into manufactured workpieces and parts based on: mathematical rules and methods, consideration of macro and micro geometry, possibilities for measuring of quantities (especially tolerance quantities) and evolution of uncertainty, etc (Durakbasa et al (2001)). The Geometrical Product Specification (GPS) is the modern and updated symbol language that is used for specifying the functional requirements in technical drawing (Bennich and Nielsen (2005)). It is a common tolerancing language and is used worldwide. It has been proved that GPS can save up to 15% in manufacturing cost through reducing misunderstandings and the ambiguity in defining the tolerance requirements (Humienny et al (2001)). However, the current GPS standards are too complex, abstract, and theoretical to be applied efficiently by the designers and engineers in manufacturing industry (who are often not GPS system experts). It is difficult for engineers who are not familiar with GPS matrix system to search for a single parameter among hundreds of paper based files and even more difficult to cross-refer and link them to form useful information. Not to mention the use of some inference rules to generate knowledge from the information. In order to overcome those current implementation problems, an intelligent GPS knowledge based system and a corresponding innovative inference mechanism have been developed, which led to development of an integrated GPS knowledge platform to facilitate rapid and flexible using of GPS. During the development of the virtualGPS knowledge system, the first stage is focused on finding a way to model the GPS knowledge base and developing a database to persist it. Researchers of this project devised a categorical object model based on Category Theory and developed an object-oriented Database Management System (DBMS) that can fully support this object model. The detailed rationales for justifying the research and development of the categorical object model can be found in a separate publication (Xu et al (2007)). Another publication can be referred to give detailed introduction on the virtualGPS system (Xu et al (2008)). This paper only focuses on giving a short evolution on the categorical DBMS with current mainstream relational DBMSs and object-oriented DBMSs. This paper can also be used as a guide for readers on choosing suitable DBMSs for their advanced knowledge/information systems relating to aerospace, manufacture and biology areas.

2 DATA MODEL COMPARSION

As the categorical DBMS is devised to support the categorical object model, the first evaluation part should be focused on the comparison of the categorical object model to other two data model that are widely used at present: relational data model and Object Database Management Group (ODMG) object model. Nowadays, there are around 40 relational DBMS products developed by various vendors (e.g. Oracle, SQLSever and MySQL), which have dominated the market for last three decades. The most important reason for the success of relational DBMSs is that they have a universal formal basis − relational data model based directly on Set Theory. The current ODMG standard 3.0 for object-oriented DBMSs has been adopted by several mainstream object-oriented DBMSs at present such as Objectivity, Versant and ObjectStore (Cattell et al (2000)). Table 1 demonstrates a comparison of these three data models in aspects of modeling capability and mathematical support.

Based on the Table1, the key feature for relational data model can be highlighted as:
The relational data model has good mathematical foundation which gives users a clear and formal construct ("Table") to model data and a rigor manipulating mechanism based on relational algebra and calculus on sets. However, it is weak in modeling of complex object structures especially for modeling of multi-level constraints/mappings and object nests.

The key feature for ODMG object model can be highlighted as:
The ODMG object model has good capability of modeling complex object structures but it is lack of mathematical foundation. So it is difficult to ensure the integrity and consistency of database schema after any manipulations such as deletion, updating or addition. This also becomes the main problem for development of object-oriented DBMSs.

Table 1 shows the categorical object model can satisfy both objectives − having good capability of modeling complex object structures especially in handling the multi-level constraints and mappings while offering a rigor mathematical foundation based on Category Theory, like the relational data model based on Set Theory. It provides an uniform way to model both static (attributes) and dynamic (methods) aspects of objects by using different types of arrows. Moreover, it can define a manipulation language based on functor mappings and compositions, so the integrity and consistency of database schema can be ensured through diagram chasing and algebraic deduce. This is the main reason why researchers of this project want to devise an object-oriented DBMS based on the categorical object model. Once the complex GPS knowledge base is clearly been represent in categorical object model diagrams, database developer can directly store them into categorical DBMS without requiring to grogram any mapping codes between the data in the database and the data in the application. The following section will give a test case to evaluate the categorical DBMS.

3 EVAULATION WITH RELATIONAL DBMS

In the verification step of this system, system users can verify measured values of products with tolerant values of GPS parameters suggested by their specification component of the system. In order to support this function, the DBMS should have the ability to store the measurands, measured values, comparison related information and comparison results for further queries. A test case is defined here to test the categorical DBMS. The surface texture knowledge base of the virtualGPS system suggests that the measurand for a cylinder liner is the surface parameter *Rz* with a tolerance of 4 µm. Table2 lists the measured values of *Rz* calculated on a manufactured cylinder liner.

The comparison information contains the comparison rule − "*max-rule*" (where the requirements specify a maximum value of the parameter, none of the measured values of the parameter over the entire surface can exceed the suggested tolerant value.), the measurement instrument (revolution, space), and the comparison result etc. Figure 1 shows the comparison procedure.

The categorical view of the comparison procedure is illustrated as Figure 2:

In Figure 2, $\boldsymbol{F_1}$ and $\boldsymbol{F}_2$ are functors mapping from class category "*Measurand*" to class category "*Value*". The σ is natural transformation mapping from $\boldsymbol{F_1}$ to $\boldsymbol{F}_2$. The $\boldsymbol{F_1}$, $\boldsymbol{F}_2$ and σ form a natural transformation square. The natural transformation σ ensures the diagram commutes as defined in the Category Theory, which is that two paths from the values for domains of arrows in the source category $\boldsymbol{F_1}$ ($dom(f_i)$) to the values for the codomains of arrows in the target category $\boldsymbol{F}_2$ ($cod(f_i)$) should be equal. In this case, the natural transformation square is used to link the suggested measurement pairs (GPS surface texture parameters to tolerant values suggested by specification part) to measured pairs (measurands to measured values inputted by users). Keeping these multi-level mappings in the database is very important, because it is useless to store only comparison results for verification without knowing the corresponding suggested measurement pairs and measured pairs. Figure 2 also shows a 2-ary pullback relationship structure between the natural transformation square and a class category "*Comparison*". The "*ComparsionResult*" is a class category holding all information generated from this relationship link (e.g. comparison result, resolution of measurement instrument, traverse range of measurement instrument, measurands and measured value). Using the instance categories of the "*Measurand*", "*Value*", "*Functor*", "*NaturalTransformation*" and "*Comparison*" as input of the verification inference engine (e.g. max-rule) in the virtualGPS, the final comparison result together with related comparison information will be stored in the instance categories of the "*ComparsionResult*". All arrow mappings, functor mappings, will be preserved and all constraints (e.g. the parameter type in source side of natural transformation σ must equal to target side) will be checked.

To implement this case in relational DBMS, the first problem is that it is impossible to store dynamic data structures in relational DBMSs. The data structures that are dynamic means they can grow and shrink while computing programs are running. Table 3 shows differences between static and dynamic data structures.

Because the system can not know how many measured values will be inputted by users in advance, dynamic data structures are suitable choices for holding data. Hence, the capability of storing and retrieving dynamic data structures is important for the implementation. In the categorical DBMS, the class category "*Value*" that extends a dynamic data structure "*CTTree*" can be used to store measured values. The second problem is that the relational DBMSs are difficult to record the natural transformation mappings because they are break normalization rules, so constraints will be lost during persistence.

The main problem for other object-oriented DBMSs to implement this test case is that they can not directly support the categorical object model. So different database developers may have different ways to define classes, which is an error-prone process. As objects in database are different from objects in application, misunderstandings may occur between GPS knowledge base designers and the database developers. Because of the absence of a multi-level mapping constructs in traditional object-oriented DBMSs, the multi-level constraints can be easily lost during persistences. Moreover, in other object-oriented DBMSs, they often directly involve one class into another class to form a relationship between the two classes. This has three problems:

- There is no a class definition to hold the information generated from the relationship link. Therefore, queries on the relationship information are difficult to get. The query closure is also difficult to achieve without a formal relationship structure.
- It is difficult to check the cardinality and membership for a relationship. This also leads to the complexity of updating or deleting objects involved in a relationship from the database.
- Break BCNF normalization rule.

Table 4 gives a comparison of the categorical DBMS with other object-oriented DBMSs (Objectivity9.0 and Versant7.0 in this paper).

Based on Table 4, the advantages of the categorical DBMS for other mainstream object-oriented DBMSs are highlighted here:

- Design and implement a categorical object model. This categorical object model can map complex object structures into mathematical formalizations in Category Theory. Thus, Algebras and calculus defined in Category Theory give a formal mathematical background to ensure integrity of database schema and defining constrains or operations.
- The categorical object model is especially good at representing the multi-level architecture, which enable advanced constraint specifications and good extensibility of designed models.
- The algebras and calculus such as arrow composition, arrow mapping, functor composition and functor mapping can be used as basis for implementation of a query language with closure.
- Involving integrity checking mechanism in both intra and inter category levels. Thus, the integrity and consistency of the database schema can be ensured in the categorical DBMS.

The categorical DBMS is not intended to support more database concepts than other DBMSs. Rather it aims give a formal mathematical basis for modern object-oriented DBMS, and it should fully support the design and implementation of the virtualGPS in Java.

At present, the main shortcoming of the categorical DBMS is that it is heavily depending on the Java language. Therefore, database developers for the categorical DBMS must know Java programming.

Researches of this project try to devise an Object Definition Language (ODL) that is independent of any real programming language based on the ODMG standard 3.0 to alleviate this shortcoming.

4 CONCLUSIONS

This paper focuses on an evolution of the categorical DBMS that forms a core part of the virtualGPS system. The categorical DBMS is a prototype to prove the applications of the Category Theory based modeling. Although the categorical DBMS is not the full-fledged DBMS that can be compared with other commercial DBMSs, we have demonstrated that the categorical DBMS can gracefully store and manage the complex data structures gained from GPS standards and provide consistency of database schema.

REFERENCES

BENNICH, P. and NIELSEN, H. (2005), *An Overview of GPS – A Cost Saving Tool*, Institute for Geometrical Product Specifications, First Edition.

CATTELL, R.G.G., et al. (2000), *The Object Data Standard:ODMG3.0*. ODMG, Morgan Kaufman, San Francisco, CA.

DURAKBASA, N.M., OSANA, P.H. and AFJEHI-SADAT, A. (2001), *A general approach to workpiece characterization in the frame of GPS(Geometrical Product Specification and Verfication).* Machine Tools & Manufacture, Vol. 41, pp.2147-2151.

HUMIENNY, Z. (2001) *Geometrical Product Specifications course for Technical Universities*, Warsw BIELSKO-Biala Erlangen Huddersfield Tallinn Vienna, Huddersfield.

XU, Y., XU, Z. and JIANG, X. (2007) *Machining Surface Texture Knowledge Management Using a Category Theory-Based Object-Oriented Database*, System and Information Science Notes, Vol.1, No.1, pp.83-88.

XU, Y., XU, Z. and JIANG, X. (2008) *Category Theory-Based Object-Oriented Data Management for Web-based Virtual Manufacturing*, International Journal of Internet Manufacturing and Services, Vol.1, No.2, pp.136-159.

	Relational data Model	**ODMG Object Model**	**Categorical Object Model**
Modeling Capability			
Formal relationship structure (including n-ary)	YES (Based on the Descartes in Set Theory)	NO	YES (Based on the product construct in Category Theory)
Trees/Collections/Arrays	NO	YES	YES
Inheritance	NO	YES	YES
Aggregation	NO	YES	YES
Multi-level mappings	NO	NO	YES
Object nests	NO	YES	YES
Mathematical Support			
Manipulations	YES (Based on set operations, algebra and calculus)	NO	YES(based on arrow mapping, arrow composition and functor composition)
Methods/Dynamic Constraints	NO	YES(Based on Object Definition	YES(Based on method arrows)

		Language without mathematical support)	
Normalization	YES (Based on functional dependency checking on sets)	NO	YES (Based on arrow composition checking on categories)
Referential Integrity	YES (Based on foreign key definitions)	YES(Based on object identifiers)	YES(Based on initial internal objects of categories)
Memebership/cardinality	YES(by labels)	NO	YES(by typing functors)

Table 1: Comparison of three data models

Cylinder liner	***Rz* (µm)**
No.1	3.245
No.2	3.132
No.3	3.675
No.4	3.565
No.5	3.175
No.6	3.432

Table 2: Surface parameter *Rz* calculated on manufactured cylinder liner

	Static Data Structures	***Dynamic Data Structures***
Size	Size is fixed when declared	Size is not fixed
Storage efficiency	Inefficient storage (e.g. a partially full array, but space has been allocated for the full size)	Efficient storage(e.g. space can be allocated as a partially full linked list needed)
Flexibility of update	Inflexible(e.g. if one more value needs to be added past the maximum size, the array needs to be redeclared and populated)	Flexible(If one more value needs to be added past the maximum size, the linked list increases automatically)
Execution speed	Faster execution	Slower execution

Table 3: Static vs. dynamic data structures

	Categrical DBMS	**Object-oriented DBMSs**
Structures		
Formal relationship structure (including n-ary)	YES	NO
Trees/Collections/Arrays	YES	YES
Inheritance	YES	YES
Aggregation	YES	YES
Multi-level mapping	YES	NO
Rules		
Normalization Support	YES(without atomicity rule of 1NF)	NO
Referential Integrity	YES	NO
Membership/cardinality	YES(by typing functors)	YES(by labels)
Manipulation		
Algebra/Calculus	YES(based on arrow mapping, arrow	NO

	composition and functor composition	
Declarative Query	YES	YES
Closure	YES	NO
View	YES	NO
Methods	YES	YES

Table 4: Comparison of categorical DBMS with other object-oriented DBMS

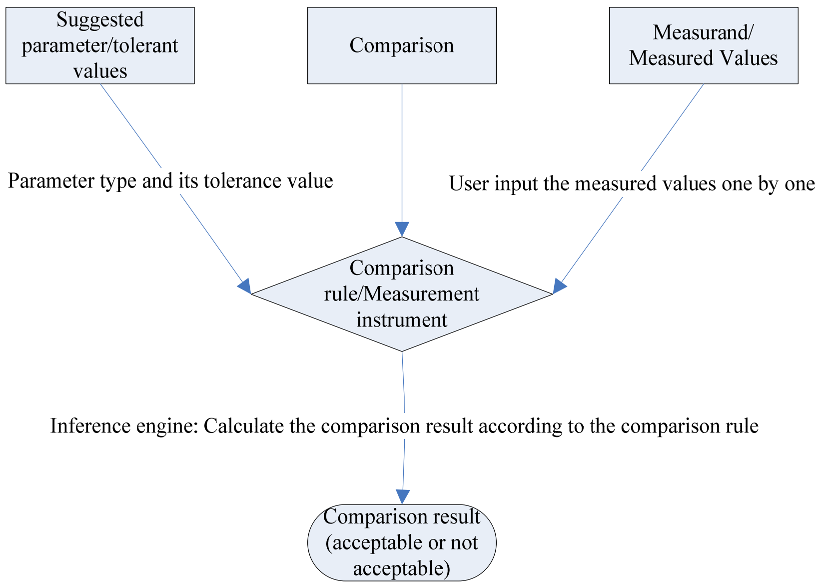

Figure 1: Flowchart of comparison procedure

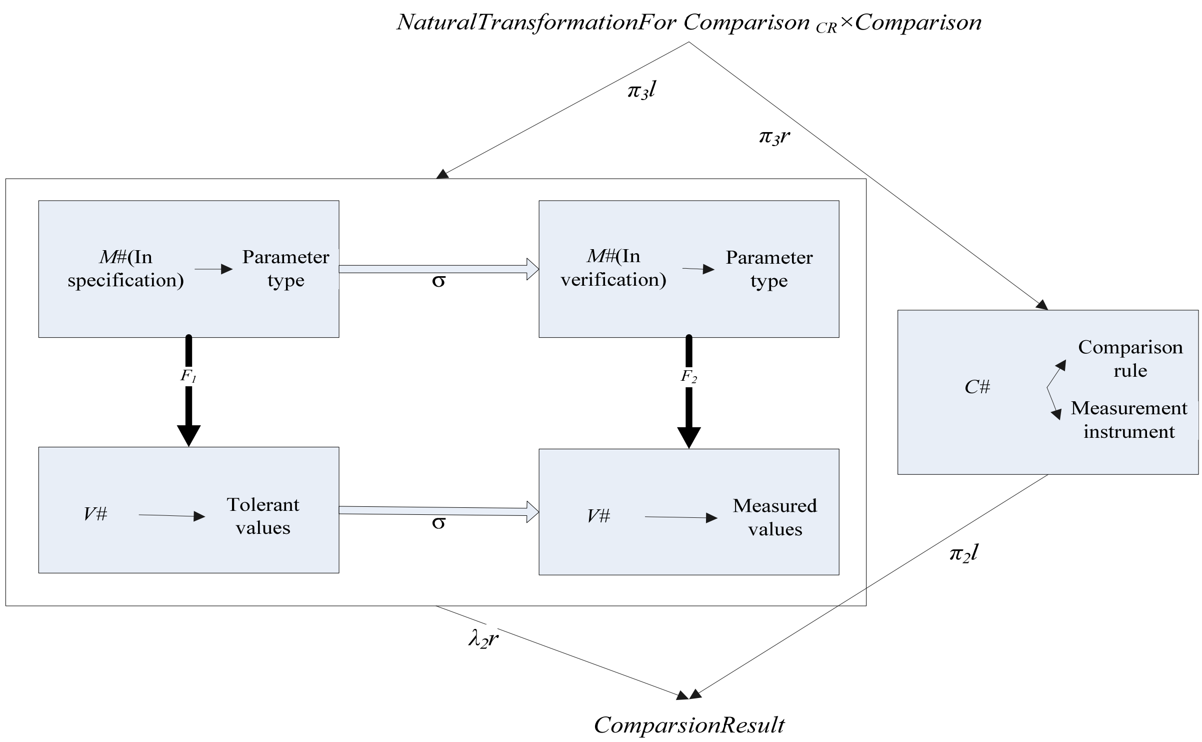

Figure 2: Categorical view of comparison procedure