Towards a Feature-Based and Fine-Grain Product Repository for Heterogeneous Computer-Aided Systems

M. M. Uddin [1], Y.-S. Ma [2]

[1] Department of Mechanical Engineering, University of Alberta, Edmonton, AB, Canada, mdmoin@ualberta.ca

[2] Department of Mechanical Engineering, University of Alberta, Edmonton, AB, Canada, yongsheng.ma@ualberta.ca

Abstract

Sharing and integrating product and process information among different computer aided applications is inevitable to follow the pace of industrial competition. Heterogeneous representations of data and knowledge utilized among different applications are major barriers to achieve engineering information system integration. This paper introduces a concept of a feature based product repository where database table schemas have been designed based on a neutral format scheme and uses unified features to maintain consistent relationship with parts' application specific features. The repository is designed to store nongeometric engineering information in addition to traditional geometric information. This paper contributes to the framework design of the proposed neutral repository system and the significance is that the neutral data structures and the fine-grain information modeling method help overcome the bottleneck interoperability problem of engineering systems.

Keywords:

Fine-Grain Product Repository; Unified Feature; Engineering Informatics

1 INTRODUCTION

Organizations have been using various computer aided tools for different product lifecycle stages to make the process faster in realizing the end product, such as computer aided design (CAD), process planning (CAPP), manufacturing (CAM), engineering analysis (CAE) and other tools (CAx). Besides, many organizations work collaboratively for product development and manufacturing, and they usually do not use the same software tools. This situation makes integration or sharing of product and process information among different systems inevitable. The heterogeneous representations of data and knowledge utilized among these applications create a major barrier to achieve the integration among such systems.

Feature based application has the potential capability to integrate and share necessary information among CAx applications [1]. Traditionally, partial transfer of product information has been achieved by translating from one proprietary format to another or extracting and converting features from one system to another. For example, features are extracted from CAD models for process planning. These approaches involve flat files that store the product information in ASCII or binary format. Such flat files suffer from drawbacks of data redundancy, wastage of storage space, lack of data integrity and potential data conflicts. File based information sharing also has the difficulty in searching, indexing, access control and multi-view representation due to the rigid information grain size [2][3]. A database repository has the potential to overcome these limitations [4].

This paper explores the feasibility of a product repository, which is feature based and neutral in nature, to store information from different applications and to integrate them through unified features [5]. A case study has been provided illustrating the product feature modeling and database schemas for a typical design mechanism.

2 LITERATURE REVIEW

Engineering systems have not yet made full utilization of the advantages offered by the current database technology to manage the large amounts of data. Hoffman et al. [6] proposed an architecture in which a product master model is used to integrate CAD systems with downstream applications. The client subsystems such as CAD, process planning and other downstream processes deposit some of their information to the master model as well as maintain their private data repository. Thus the master model contains the net shape of the product and some other process information and informs the clients after any change of information is made in master model. This architecture suffers from data redundancy by storing the same information in both the master model and client subsystems. Bronsvoort et al. [1] presented a multiple view feature modeling approach to integrate product development activities, e.g. conceptual design, assembly design, part detail design and manufacturing planning. They also mentioned some mechanisms for consistency maintenance among the views. However, all of the above works are flat file based, and the information grain size is large which makes data sharing among different applications difficult. Sun et al. [7] proposed a methodology for building a database for reusing past design knowledge in developing new products by archiving technical documents on robotic design, but they did not consider product information from sketches, drawings and CAD models.

Researchers at NIST [8] developed representations for core product knowledge which are then implemented in a design repository to store design attributes in relational databases for web-based collaboration among product development teams. Xue el al. [9] reported a concurrent engineering oriented design database representation model (CE-DDRM) by incorporating NIST's Function-Behavior-Form based design modeling approach. The model represents the concepts of database components in three levels, i.e. concepts and behaviors of modeling components, generic design libraries, and specific design cases. Bohm et al. [4] described the development of a data schema to archive basic design concepts into a repository called UMR Design Repository to support reuse of design knowledge in concept generation and risk assessment. The repository is artifact centric and designed by a PostgreSQL relational database. Although these works used databases as the repositories for design related properties for standalone tools, but they are not integrated across different other applications such as CAD, CAM and process planning. Their geometric information is still kept as CAD files linked to the repository. Therefore, the information grain size is coarse (in fact

the whole file) that the features and lower-level geometric information entities cannot be accessed or modified within database. Since those repositories are not feature based, feature level interoperability cannot be achieved. Kim et al. [10] proposed a CAD data exchange method to maintain design intents captured by model history, features, parameters and constraints that justifies features' use to share high level product information. Zhou et al. [11] discussed a STEP enabled generic product modeling framework which uses STEP exchange file, VC++ working form and DBMS to store product model. But the database does not store product lower level geometric information. Regli et al. [12] discussed the challenges of interpretability of digital geometric model such as file formats, logical object encodings, object metadata and organizational workflows over the long lifespan. The authors observe that potentially, the data types and data structures of a SQL database could be interpreted easily by diverse applications.

In this work, the authors propose the design of a product repository implemented by SQL database which allows feature level interoperability of product data based on an extended STEP framework and represents both nongeometric and geometric information of the product.

3 PROPOSED APPROACH

Ideally, database repository can overcome the limitations of flat-files; stores and retrieves data, information and knowledge at various levels of abstraction, including low-level geometric information of components, sub-assemblies and assemblies, feature information, as well as other non-geometric information such as materials, functions and designer's intents. Therefore, a unified feature-based fine-grain product repository [5] has been proposed as a solution to the problem of interoperability among engineering applications. With a cluster of well defined generic features, a multi-view product database is defined generically and brings together the attributes and methods of all the supported applications. Theoretically, unified feature-based database is capable of removing redundant information and establishing persistent relations among different feature elements. Different application feature models for the diverse engineering aspects capture the vertically organized product information in the form of specific application features which are mapped to neutral application feature model to support data sharing among different applications.

Ma et al. [3][13] have introduced a fine-grain and feature based product database; neutral features can represent the complete product data model and be implemented in a commonly available SQL database via its API. Feature relationships are then mapped into database schemas. The reported effort needs more research because the tables in the database have to be created manually which restricts the creation of new user defined feature types in the database. Further, the validation of feature types and propagation of changes cannot be achieved. The authors continued the effort [3][13] to design a product repository storing fine grain product information, both geometric and semantic, based on a neutral format and unified features.

Based on the object-oriented approach, product and process attributes and methods of different application features generated by heterogeneous systems can be integrated through the structure of a database repository. The system architecture of the proposed product repository has been shown in Figure 1. The system consists of five layers. The top layer is specific application tools or modeling systems. Generally, a user interacts with the system making use of certain application tools such as CAD, CAM or CAE systems to generate application models. Thus this layer provides the user scope to provide application specific input to the repository system. The second layer is application programming interfaces (APIs) that gives the facility to capture geometric information as well as design intent using user-defined attributes and features along with the traditionally available features. The APIs can be used to customize the application tools according to user needs and to incorporate additional engineering knowledge and rules to cycle the geometric and feature information for the feature based integration.

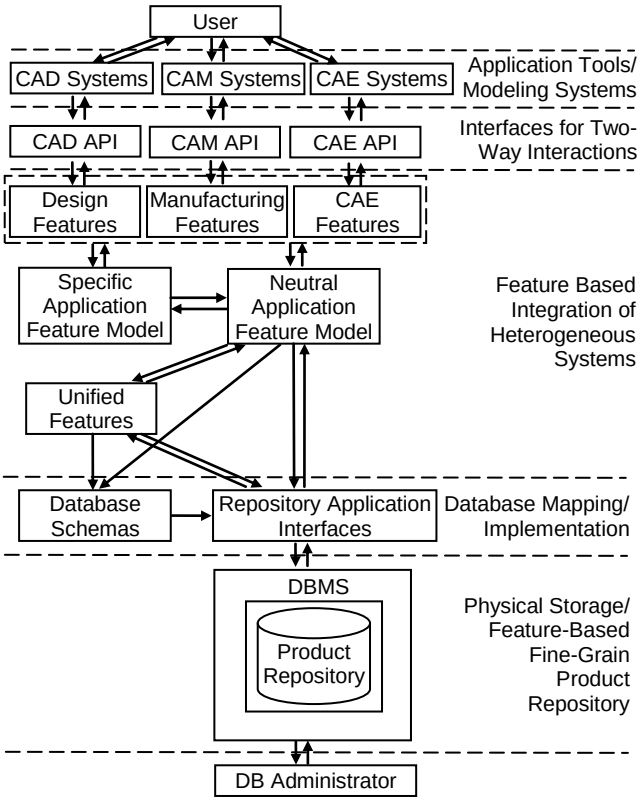


Figure 1: System architecture of the proposed repository

The third layer supports feature based integration of heterogeneous systems by taking the output from the API applications and creates specific application feature models, e.g. design features, manufacturing features and so on. The specific application features are then mapped and associated to a neutral application feature model on the basis of an extended STEP framework which can be similar to STEP 224 for machining features, and thus facilitates interoperability among different applications. Unified features support the neutral application feature model in a generic form in order to establish relationship among application features from different tools/domains to achieve integration and to maintain consistency by propagating changes made in one application to the related entities at different levels. In the fourth layer, database representation schemas are created according to unified feature types and neutral features to store product data in the repository via database interfaces. This work adopts a relational database to represent the neutral format to maintain consistent product information in the repository. Those schemas are then implemented using database application interface, e.g. MySQL API. The application interface works as a common development environment for creating the functions to pump the product information into the repository and retrieve from it.

The bottom layer in the system is the physical storage. The granularity of product information is scalable from the assembly, to

the high level features and further to the low level topological and geometric entities. Thus the repository completely represents product model to serve as a hub of all types of information. It is the authors' intention to support multi-view feature domains and generic process modeling approach, like IDEF; but due to the space limitation, this aspect is left out for future work. Once the repository is set up, the multi-facet system integrity has to be maintained by a dynamic database agent system. Since all of the information is now in the repository, different sets of information for specific purposes can be retrieved, modified or updated based on the user profiles. Ideally, from modifications done in the repository, the agent system can update the CAD models, CAE models or other models reflecting the changes in an associative manner. The agent system is another future research topic.

4 FEATURE MAPPING AND ROLE OF UNIFIED FEATURE

4.1 Mapping between application feature and neutral feature

Feature modeling has become popular in using CADCAM tools; features can potentially act as generic information units and carry enormous engineering information in addition to geometric and topological information. Such advanced use of feature is not common yet in available engineering systems because in different CADCAM systems, proprietary feature definitions are used. Mapping from specific application features to a neutral format is necessary in order to share feature level information. In this work, STEP 224 is referred to represent manufacturing features. Currently, design feature representation is not yet standardized. The design features used in this work have been taken from the representation of CSG primitives and manufacturing features of STEP and commercial software NX. For example, the mapping of slot design feature from NX to STEP format is shown in Figure 2.

It is seen that the location of origin and orientation of coordinate axes are at the middle of the top face of the slot in NX whereas it is situated at the bottom face and at one end of the STEP feature. *slot_direction* is defined by y-axis in NX, but by z-axis in STEP. NX defines rectangular slot mainly by three parameters, i.e., *width, depth* and *distance* whereas STEP uses a *sweep_shape* which is an open profile and a *course_of_travel*. Thus *width* and *depth* must be mapped together to *sweep_shape* in STEP and *distance* to *course_of_travel*. It seems clear that considerable works are to be involved in the mapping from one application feature to neutral feature and vice versa.

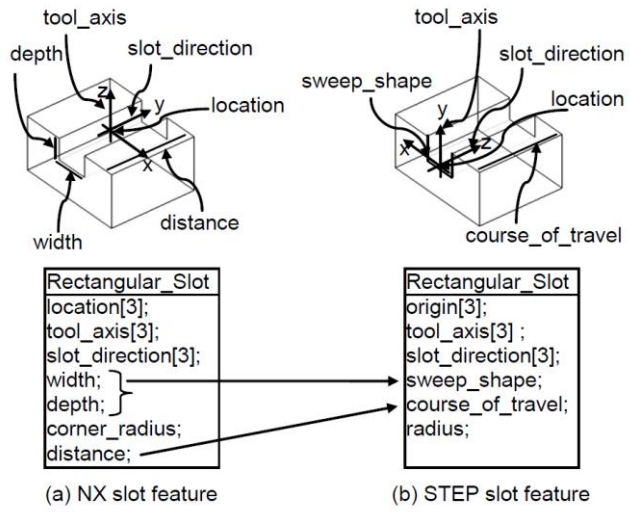


Figure 2: Mapping between NX and STEP feature

4.2 Relations between unified feature and application feature

The unified feature scheme generically defines the common attributes and methods of application features and establishes relationship among them by keeping references to the relevant object entities. A detailed discussion on unified feature definition and change propagation can be found in [5]. An example of attributes and methods defined in unified feature and the relationship with application features are shown in Figure 3.

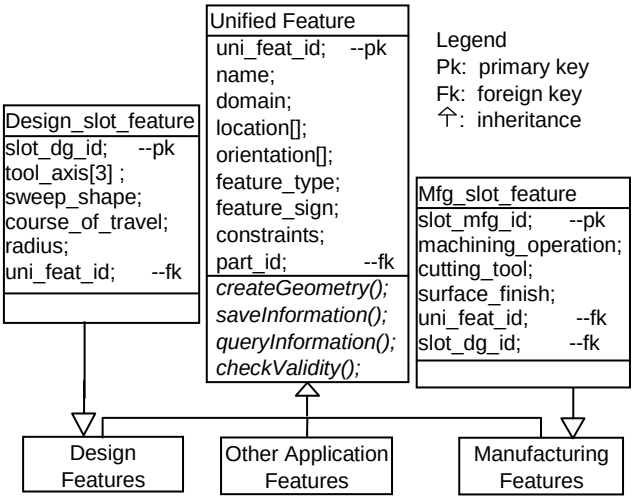


Figure 3: Relationship of unified feature with application features

The application features inherit the generic attributes and methods of unified features as subclasses. In the case of database implementation of features, the members of unified features and specific application features are kept in separate tables and the entries in unified feature table are referenced by using foreign keys (fk). It thus avoids duplication of the same data in several tables. In Figure 3, definitions of slot design feature and slot manufacturing feature are shown. Both of them reference to the corresponding unified feature information by using *uni_feat_id*. Associations between different domain features can be supported by one or more foreign keys. For example, if a manufacturing slot feature uses the same geometry of a design slot feature, it can be realized by referencing *slot_dg_id* as a foreign key in addition to its own attributes such as machining operation, cutting tool, and surface finish to complete its definition.

The methods defined in unified feature class are *createGeometry(), saveInformation(), queryInformation() and checkValidity()*. They are generic virtual functions suitable for all application features. *createGeometry()* retrieve data from the database and uses *create_api()* function to generate the application models; in which, for this slot example, NX API *uf_modl_create_rec_slot()* can be used for NX feature creation. *saveInformation()* cycles attributes from application models and stores into database tables. *queryInformation()* executes query to get feature specific information from the database. *checkValidity()* verifies whether the data satisfies the constraints specified such as data type, relation between variables, and generation of valid application models and propagates any change made by the user. More similar methods can be defined within the unified feature class definition.

5 DATABASE SCHEMA OF FINE-GRAIN PRODUCT REPOSITORY

Database schema defines what types of product information can be stored into the database tables and the relationship among the tables to represent the product completely in the repository. Fine-

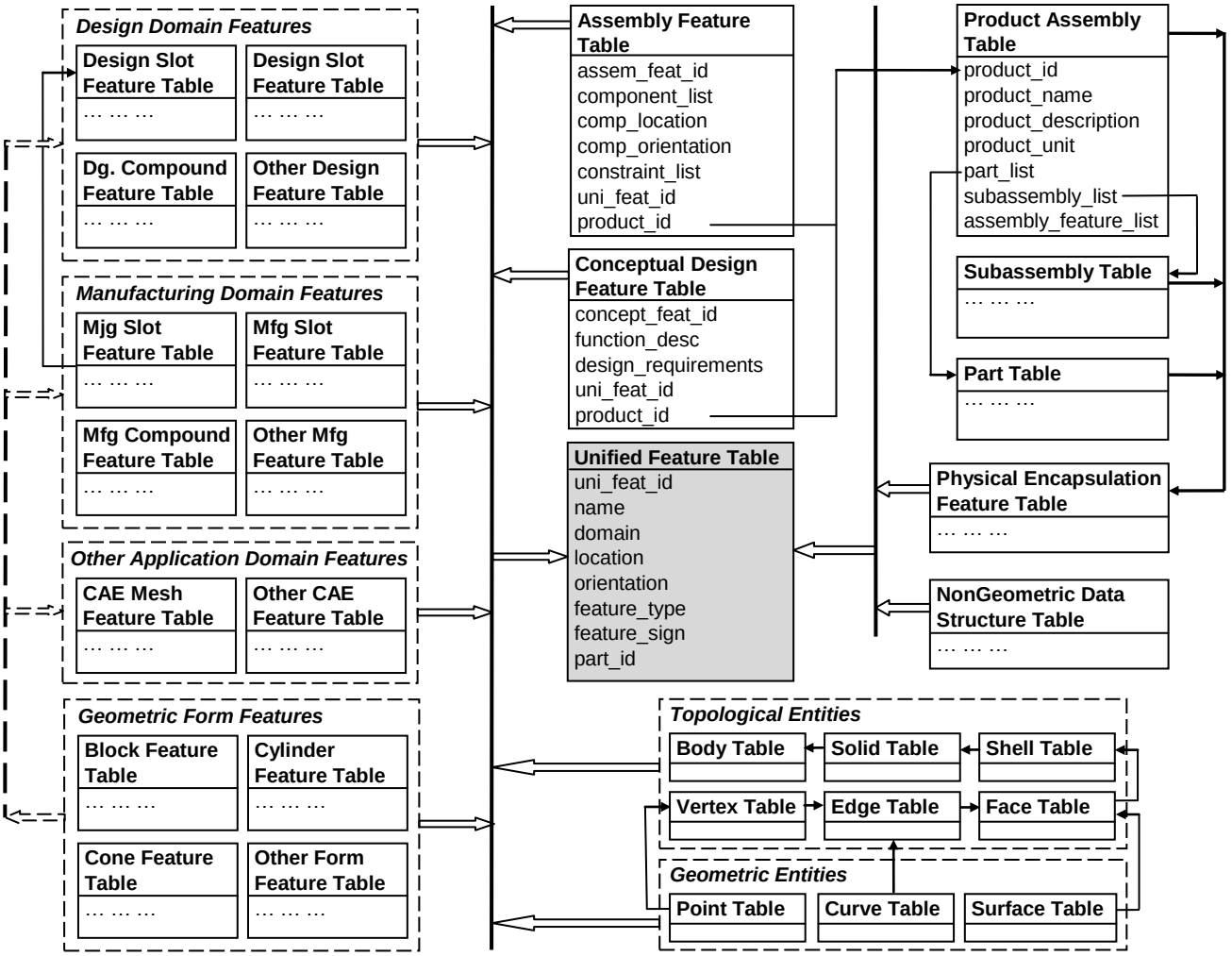


Figure 4: Partial schema of database tables for the fine-grain product repository

grain product information means that the granularity of information is small enough to store, index and access to the lower level topological and geometric entities so that the repository does not depend on external applications for the actual geometric data manipulation, validation, and inquiries. The design of database schema for the proposed fine-grain product repository is shown in Figure 4. The product repository design has referred STEP framework.

The elements of the database tables and their data types are defined as a neutral data structure and based on unified feature definitions. A table can have only one primary key (*pk*) which uniquely references to a particular record in that table. A primary key can be used as a foreign key (*fk*) in another table to establish a parent-child relationship between tables. The product repository has been designed based on features and has different levels to represent a product fully. The three tables, i.e. *product_assembly_table*, *subassembly_table* and *part_table*, together represent the assembly tree structure of a product. The product assembly table identifies every product uniquely using primary key, *product_id* and stores those attributes which are related to the overall product description. With each new entry into the *product_assembly_table*, the repository will store all the data needed by the subsequent tables. The components of an assembly comprises of *part_table* and *subassembly_table*. A subassembly consists of parts and may also have another subassembly which can be denoted by referencing to the *parent_subassembly* using *parent_subassem_id* field. *Product_assembly*, subassembly and part are connected to *unified_feature* table through *physical_encapsulation* feature table. Physical encapsulation features capture the engineering attributes of a product such as functions, design intent, and material requirements.

The *assembly_features* and *conceptual_design_features* make another level in the database. Assembly features represent the component list, their locations, orientations, configurations of components, and constraints applied to the components. Conceptual design features are used to capture design requirements such as materials, strengths, surface finish and initial design output. Geometric form features are the basic shapes which are used to model a CAD part using Boolean operations. Application domain features form another level which has different sublevels such as *design_domain_features*, *manufacturing_domain_features*, and *other application_domain_features*. The design features are those features using basic geometric features such as *slot, hole, pocket, chamfer, blend, extrude, sweep, free-form* features etc. The manufacturing features may use the same design features for representing the shape of part referenced by foreign keys and include some additional information such as machining operation, cutting tool, surface finish required etc. The geometric form features

support design domain, manufacturing domain and other application domain features.

Each of these features represents feature level information of a product and has its own specific attributes and parameters, and thus can be implemented using a database table. These features share the common attributes and methods defined in the unified feature by referencing to it using *uni_feat_id* foreign key. The unified feature relates the features to the part they belong to by referencing back to the part entry in the *part_table*. The methods of unified features can be used to send SQL statements to the server to activate triggers in the associated table when a particular event occurs for that table such as to check the validity of the values entered.

The lowest level in the repository deals with the topological and geometric entities of the product that actually represents the shape and nongeometric data structures. The topological entities include body, solid/sheet, shell, face, edge, and vertex. Topological entities have their corresponding geometric entities such as surface, curve and point respectively. The low level information is referenced by the corresponding features and parts to which they belong to. Thus the repository stores all the detailed data of a product which is named as fine-grain product information. The unified features link the features from different domains to the part table and eliminate redundancy of information.

6 PRODUCT MODELLING USING API FOR THE TWO-WAY INTERACTIONS

The proposed repository requires cycling feature and geometric information from the application systems to store it and also to regenerate application model by taking information back from the repository after any modification is performed by the user. Thus the system needs an interface for two-way communication between application tool (such as CAD software) and the repository. In this work, NX Open C API is used to cycle information from NX proprietary files. The API programming also offers users more control on the CAD system to customize it depending on user requirements such as defining user defined features, and user defined attributes. Figure 5 illustrates code snippets showing API functions for cycling product information from NX CAD model, pumping them into repository and using user defined attributes to capture product information such as part material and designer's intent.

```
//cycling product info from CAD model
t_part = UF_PART_ask_display_part();
t_root_occ = UF_ASSEM_ask_root_part_occ(t_part);
UF_OBJ_cycle_objs_in_part(t_part,UF_ftype,&feat);
...  ...  ...
//storing product info into database
mysql_real_connect(conn,"ma-server","moin",
            "passwrd","moin_part_db1",0,NULL,0);
sprintf(msg,"INSERT INTO Body_Tab(body_id,
            part_id) VALUES(NULL,'%d')",part_id);
mysql_query(conn, msg);
...  ...  ...
//recreating part model/attributes
UF_PART_new("d:\\nx_parts",1, &t_part);
UF_MODL_create_block1(NULL,origin,edges,&t_blk);
UF_MODL_create_rect_slot(loc,tool_axis,dir,width,
depth,length,face_link,nullt,nullt,&t_slot);
...  ...  ... UF_ATTR_value_t
att_value;//attribute data struct
att_value.value.string="AISI 1040";//attrib value
UF_ATTR_assign(t_wpart,"part_materl", att_value);
```

Figure 5: Code snippets to cycle, store and recreate product model

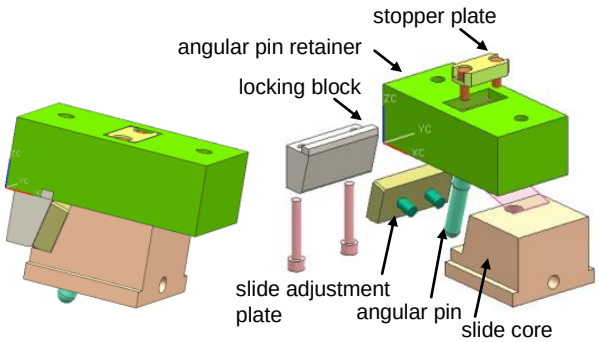
The product modeled in this work is a slider mechanism of plastic injection mould assembly. The slider mechanism is an assembly of subassemblies and parts with twelve parts in total. These parts have many interdependencies in terms of locations, orientations and dependent parameters. This is why, the modeling has been carried out parametrically using expression file to capture all the parameters required and to establish relationship among parameters. Expression file is especially useful to calculate parameter values from complicated equations. Figure 6 demonstrates partial view of an expression file that shows part location relationship in an assembly, part driving parameters, detailed parameter values and dependencies for groove feature.

```
// assembly configuration expressions
AP_pos_x=APR_pkt_loc_x
AP_pos_y=APR_pkt_loc_y
AP_pos_z=(APR_pkt_loc_z-APR_pkt_height)
/*part position is associated to the
accommodating feature on another part;
here an angular pin is located according to the
pocket on the retainer where an assembly relation
is created */
...  ...  ...
//part driving expressions for "Stopper Plate"
SP_W_width=16.0
SP_L_len=32.0
SP_P_centr_to_centr=23.0
...  ...  ...
// detailed feature expressions
AP_D_cyl2_dia=10
AP_rect_groove_dia=(AP_D_cyl2_dia-1)
AP_rect_groove_width=2
```

Figure 6: Partial view of an expression file

7 CASE STUDY – SLIDER MECHANISM

A case study of slider mechanism which is a part of plastic injection mould assembly has been chosen to model the product using CAD API and to implement the product repository system. The assembly and its exploded view with the subassemblies and parts are shown in Figure 7.



a) Assembly view b) Exploded view

Figure 7: Slider mechanism modeled using NX Open API

The slider mechanism has been modeled by adopting both top-down and bottom-up assembly approaches. It has two subassemblies such as *stopper plate* subassembly and slide *adjustment plate* subassembly. In the product repository, the product data is stored by implementing product assembly table, subassembly table, part table and some of the feature tables related to the design domain. Database inquiry results are reproduced in Figure 8 showing SQL commands and the associated database contents. Product assembly table,

subassembly table and part table together stores the product tree structure into the repository. Unified feature table (as shown in Figure 8) keeps all the features of the product and relates the features with their related view using domain and part_id. Here the features are for design view of stopper plate piece part (part_id = 3). The slot table shows some of the parameter values of slot features. These specific feature tables are connected to geometric entity tables to store lower level geometric data. Thus the repository represents a product completely by containing its feature level information and lower level geometric information and does not depend on any external CAD applications. Multiple views of the product can be achieved by extracting view specific feature and geometric information from the repository and regenerating the view model using API functions. Similarly, different applications can be integrated by first pumping product information from one application to the repository and then recreating it in another application by extracting information from the repository. In the prototype system, only features from design domain have been implemented. Features from manufacturing domain and other application domain are yet to be implemented.

This work is only at the very preliminary stage. Until now, the functions of extracting product information from the CAD model and storing the information into database have been achieved. The extraction of database contents to recreate CAD model has been achieved partially.

```
Select part_id, name, description from Part_Tab;
```

Part_id	Part_name	Part_description
1	Adjust_plate.prt	Adjusts slide core
2	Locking_block.prt	Limits slider movement
3	Stopper_plate.prt	Fixes angular pin
...	... ...	... ...

```
Select feat_id,name,domain,part_id from
Unified_Feat_Tab where part_id = 3;
```

Feat_id	Name	Domain	Part_id
43	Block	Design	3
44	Chamfer	Design	3
45	Simple_hole	Design	3
...	...	...	...

```
Select slot_id,loc_x,width,depth from Slot_Tab;
```

Slot_id	Loc_x	Width	Depth
1	0.0	9.0	5.5
2	32.00	9.0	5.5

Figure 8: Database inquiry results showing the parts and features stored (reproduced)

8 CONCLUSIONS

This research explores the feasibility of a fine grain and multi-facet product repository which can store complete product information, including features and the lower level geometric data, using neutral data structures. The significance of this work is that the suggested repository can be used to integrate and update associative application feature views. Thus it can integrate application feature models generated by different software tools. Since the repository is based on database technology, by designing appropriate table schemas, dangling and redundant engineering data can be avoided. The repository can also impose “need to know” view restrictions on “roles” or users to ensure data security by controlling profiles. The potential benefits of the proposed repository are envisaged as its capability to independently represent product information completely, to integrate different applications and to create multiple views of the same product. By using the agent technology, the proposed repository can control data integrity and consistency more effectively. Future works include the development of algorithms for checking product information validation and smooth change propagation in the repository using the agent technology, the representation of constraints in the database and the development of methods for generation of multi view product model from the repository.

9 ACKNOWLEDGMENTS

The authors would like to thank NSERC for its Discovery grant (355454-09) and the University of Alberta for startup grant.

10 REFERENCES

[1] Bronsvoort, W.F., Noort, A. (2004): Multiple-View Feature Modeling for Integral Product Development, CAD, Vol. 36, No. 10, pp. 929-946.

[2] Connolly, T., Begg, C (2004): Database solutions – A Step-by-Step Guide to Building Databases, Pearson.

[3] Ma, Y.-S., Tang, S.-H., Au, C.K., Chen, J.-Y. (2009): Collaborative Feature-Based Design via Operations with a Fine-Grain Product Database, CiI, Vol. 60, No. 6, pp. 381-91.

[4] Bohm, M.R., Stone, R.B., Simpson, T.W., Steva, E.D. (2008): Introduction of a Data Schema to Support a Design Repository, CAD, Vol. 40, No. 7, pp. 801-11.

[5] Ma, Y.-S., Chen, G., Thimm, G. (2008): Change Propagation Algorithm in a Unified Feature Modeling Scheme, CiI, Vol. 59, No. 2-3, pp. 110-118.

[6] Hoffman, C.M., Joan-Arinyo, R. (1998): CAD and the Product Master Model, CAD, Vol. 6, No. 1, pp. 85-116.

[7] Sun, J., Lu, W.F., Loh, H.T. (2008): Building a database for product design knowledge retrieval - A case study in robotic design database, Robotics and Computer-Integrated Manufacturing, Vol. 26, No. 3, pp. 224-229.

[8] Szykman S. and Sriram, R.D. (2006): Design and implementation of the Web-enabled NIST design repository, ACM Transactions on Internet Technology, Vol. 33, No. 7, pp. 545-59.

[9] Xue, D., Yang, H. (2004): A Concurrent Engineering-Oriented Design Database Representation Model, CAD, Vol. 36, No. 10, pp. 947-965.

[10] Kim, J., Pratt, M.J., Iyer, R.G., Sriram, R.D. (2008): Standardized data exchange of CAD models with design intent, CAD, Vol. 40, No. 7, pp. 760-777.

[11] Zhou, Z.D., Xie, S.Q., Yang, Y.Z. (2008): A case study on STEP-enabled generic product modelling framework, International Journal of Computer Integrated Manufacturing, Vol. 21, No. 1, pp. 43-61.

[12] Regli, W.C., Kopena, J.B., Grauer, M. (2011): On the long-term retention of geometry-centric digital engineering artifacts, CAD, Vol. 43, No. 7, pp. 820-837.

[13] Ma, Y.-S. (2009): Towards Semantic Interoperability of Collaborative Engineering in Oil Production Industry, CERA, Vol. 17, No. 2, pp. 111-119.