Core Information Model: A Practical Solution to Costly Integration Problems

Cheng Hsu*, Jangha Cho†, Lester Yee‡, and Lauire Rattner◊

Forthcoming in Computers and Industrial Engineering

Revised August 1994

* Associate Professor, Decision Science and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy NY 12180-3590

† Department of Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy NY 12180-3590

‡ Decision Science and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy NY 12180-3590

◊ Assistant Professor, Anderson School of Management, University of New Mexico, Albuquerque, NM 87120.

ABSTRACT

Computer-Integrated Manufacturing typically entails multiple information systems that, individually, need to satisfy their own functional requirements while, collectively, must work together and constitute an integrated environment for the enterprise as a whole. Thus, an enterprise information model is critical to CIM. A missing element in many CIM information models is the contextual knowledge that turn data resources into agents of integration. This alignment of data with knowledge are especially crucial for reference models (in the sense of, for instance, CIM-OSA [3]), which are recommended by international standards communities as an economical way to develop integrated information systems. A reference model could either be a road map or be a core model to start the development with. The problem with road map is usually its lack of specificity, and that with a core model is costliness, or being over-comprehensive. This paper presents a core model that is compact and suitable for small and medium-sized manufacturers. An integration theory based on the parallel formulation of information and decision processes is used together with other principles derived from the literature to guide the development of control and other types of contextual knowledge. The knowledge is then fully engineered to integrate with a generic, basic CIM data model developed from industrial scenarios. The complete model is presented along with its underlying methods.

Key Words: Data and Knowledge Engineering, CIM Reference Model, Information Integration, Metadatabase

ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation’s grants DDM90-15277 and DDM92-15620, and by ALCOA, DEC, GE, GM, and IBM through the Computer Integrated Manufacturing and the Adaptive Integrated Manufacturing Enterprises Programs at Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

1. Introduction: Reference Models For Enterprise Information Integration

Integration at the factory level and higher is fundamentally achieved through information systems that globally manage the data resources and apply contextual knowledge to these resources to effect synergism across the various functions in the manufacturing enterprise [1,2]. Therefore, developing a global model of data and knowledge that is both sound and practical is a key to any integration project; and yet this task in actuality can easily overwhelm even a major organization. This fact would not escape anyone's observation who has noticed the dominance of the enterprise integration industry by consulting companies and their multi-million dollar projects. Small and medium-sized manufacturers are especially hard-pressed for finding affordable solutions to their integration needs. One approach to improving the feasibility of modeling is to develop core reference models that particular enterprises can adopt and customize for their respective environments. This way, some common modules can be developed in a standard way to hold down the total cost of custom design and reduce the effort required for integration. The CIM-OSA (Computer Integrated Manufacturing Open System Architecture) project at Europe [2] is among the first to advocate such an approach. Thus, we begin our analysis for a solution with this project.

Following the pioneering effort of the United States Air Force Integrated Computer Aided Manufacturing Project (ICAM)[3], the European efforts produced the much heralded CIM-OSA model, which attempts to provide a manageable business and manufacturing environment with information supported decision making processes. It has developed generic constructs for the structured description of the enterprise system model and a general framework in designing and implementing systems. While calling for a comprehensive "particular model" (Figure 1) to further specify the information contents for each particular industry concerned, the CIM-OSA model stops short of delivering just that.

A more focused information requirements model is developed at Rensselaer [5,6,7], which provides an integration theory constructed from the literature and determines the basic classes of data and knowledge required for integrating manufacturing planning and control functions according to the theory. Its data classes lead directly to structural models for database analysis and design. Similarly, the knowledge classes and decision logic associate the set of data classes throughout the myriad of integrated functions and sub-functions, leading to production rules and assertions after the instantiation.

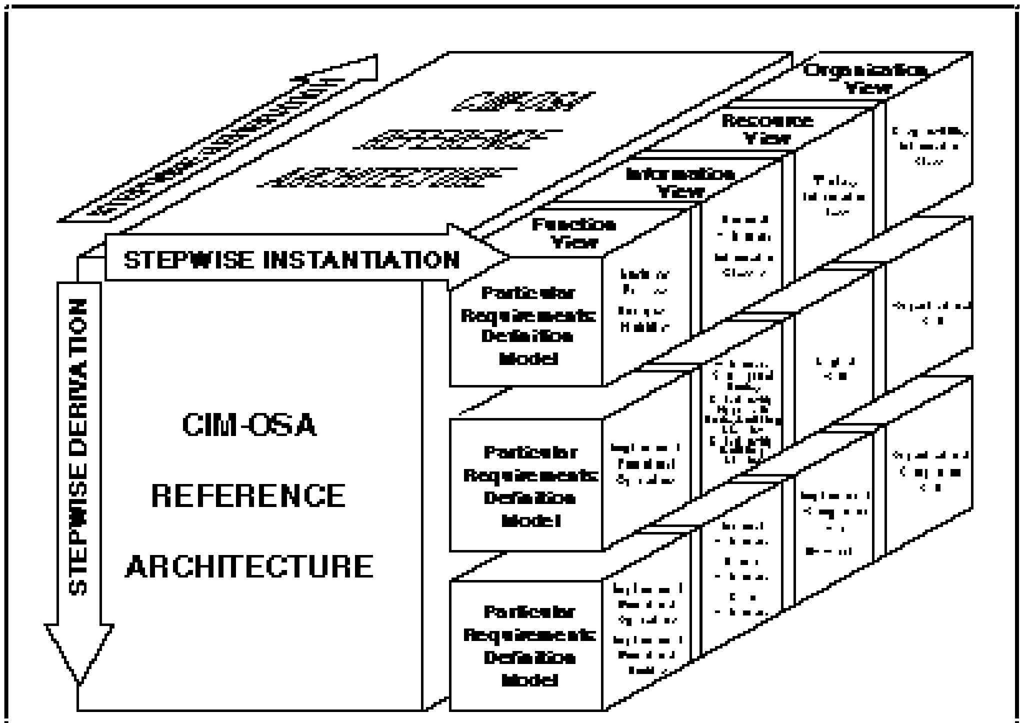

Organization View
Resource View
Information View
Function View
STEPWISE INSTANTIATION
STEPWISE DERIVATION
CIM-OSA REFERENCE ARCHITECTURE
Particular Requirements Definition Model
Particular Requirements Definition Model
Particular Requirements Definition Model

Figure 1. CIM-OSA Modeling Approach and Constructs

The resulting reference mode can be used as a road map to facilitate information planning for three cycles of manufacturing enterprises: Product (life) cycle, production cycle (based on customer orders), and part cycle (based on work orders). Figure 2 illustrates the enterprise information planning framework where (1) the reference model provides specific directions for such strategic planning goals as process re-engineering, (2) the data classes and knowledge classes of the model indicate the needs and opportunity for information systems integration for top-down IS planning, and (3) the instantiation of the reference model supplies a core checklist for bottom-up evaluation of existing functional area systems. The product cycle planning corresponds to the top-down process in the framework, while the part cycle is bottom-up and the production cycle links both. The reference model can also be instantiated into a core information model serving as a starting point of information system development in particular manufacturing enterprises (in conjunction with organization-specific details). Such instantiations, however, have not been accomplished previously.

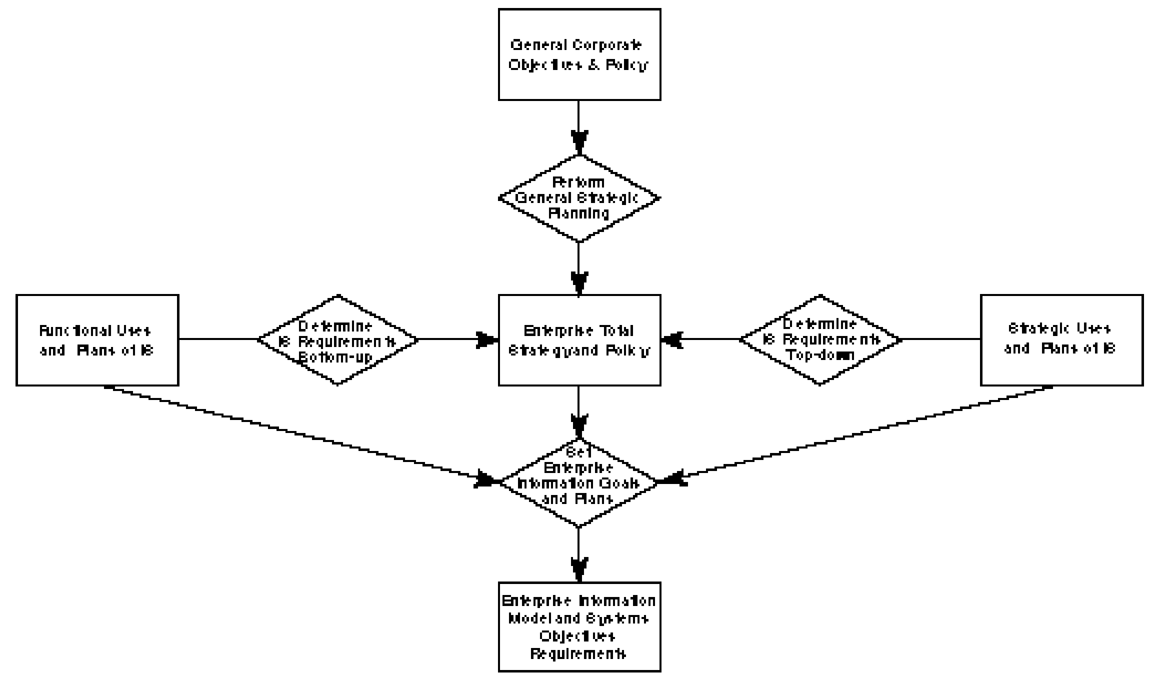

General Corporate Objectives & Policy
Perform General Strategic Planning
Functional Uses and Plans of IS
Determine IS Requirements Bottom-up
Enterprise Total Strategy and Policy
Determine IS Requirements Top-down
Strategic Uses and Plans of IS
Set Enterprise Information Goals and Plans
Enterprise Information Model and Systems Objectives Requirements

Figure 2. Use of a Reference Model for Enterprise Information Planning

Extending the previous efforts and combining them with some empirical results, this paper presents an instantiation of the reference model using generic but elaborated knowledge and database instances in the field. It also provides a methodology for applying the reference model to develop an integrated CIM information system by using a metadatabase. Both the literature and our investigations with IBM and GE[8] through Rensselaer's CIM and AIME Programs [9,10] are utilized. These results constitute the main contributions of the paper. We also consider the approach of using a core information model for CIM as proposed here novel. The core model, as mentioned above, combines data with knowledge and integrates CIM through a metadatabase - neither is considered in CIMOSA or other approaches.

The objective for this instantiated core model is soundness and compactness: including fundamental data structures and, expressly, operating rules for integration and yet remain unimposing for small to medium-sized enterprises. Finally, a conceptual framework for modeling is presented to facilitate the implementation of the design approach. This framework is employed to produce the above instantiation, and is proposed to be used for extending the core module for specific system analysis and design projects in companies. In section 2, we discuss the main sources of integration principles that guided the instantiation work, and the modeling methodology used. Section 3 presents the information model as a prototype for core CIM databases and production rules. Section 4 illustrates the application methodology and discusses the implementation issues of the model from a user's perspective. A small example showing evaluation criteria is also included. Concluding remarks are provided in section 5. In addition to the complete specification of the core CIM data and knowledge model in section 3, Appendices A, B, and C further document the model's control knowledge, data management rules, and data semantics, respectively.

2. The Methodology for Model Derivation

The unique element of the core reference model is its contextual knowledge for integration. This element is developed by using some reference frameworks including the integration theory cited above and a generic modeling method.

2.1. The Reference Frameworks for CIM Information Requirements

The literature provides many commonly cherished principles on information requirements for manufacturing integration, including those for concurrent engineering [11], design for manufacturing [12], and, of course, CIM [13]. These principles and definitions, however, usually do not provide fine enough granularity to render direct use for model development in integration projects. One could, on the other hand, attempt to derive relevant results from specific methods, models, and architecture (e.g., [14,15]); however, they would tend to be overly specific to be used for general modeling of integration. A useful source is the kind of analyses in [16]; where the facility level CIM model embodies status codes for each CIM function to control the information flow and the status changes, specifically for the MRP II, Shop Floor Control and CAD functions. This Petri-net based model provides some useful classes of the control knowledge required; however, it tracks and controls the operational sequences of sub-systems only for a *single* product part, and does not support the handling of *multiple* orders or designs. More is needed. Thus, we turn to our main source of guiding principles: the parallel formulation of an integration theory [6].

The traditional CIM formulation of standalone functions is characterized by separate responsibility and dedicated decision spaces. The information flows among these spaces tend to be sequential, amounting to a hierarchy of decision processes. These separate decision spaces could be joined and each black box process could be opened to allow the unit processes embodied within to be reconfigured with others in other boxes. That is, manufacturing functions can be organized - re-engineered - to explicitly share unified decision spaces, to configure and execute as many processes in parallel as possible - and this is the **parallel formulation** that defines integration. The justification is that by optimizing globally decision variables, each local function is managed at the maximum possible performance level while achieving synergism.

The paradigm alters the decision-making hierarchy so that peer functions operate in parallel through paired-up unit processes that are linked with necessary information flows. Since the data resources required of manufacturing functions tend to remain stable regardless of integration, the reconfiguration of processes into a parallel paradigm is achieved mainly through **contextual knowledge** that defines the processes and the information flows among them. The basic information requirements of a particular design of integration, therefore, are essentially the timing, information contents, and inter-relationships of decision processes in manufacturing functions. The factor that fundamentally determines the characteristics of these requirements is which one of the three cycles the decision processes pertain to: product development, production, and part machining. Each cycle calls for a set of required parameters for its decision processes. The interaction among cycles is also achieved through the sharing of decision spaces and alignment of decision processes in terms of timing, information contents, and flows; as depicted in Figure 3.

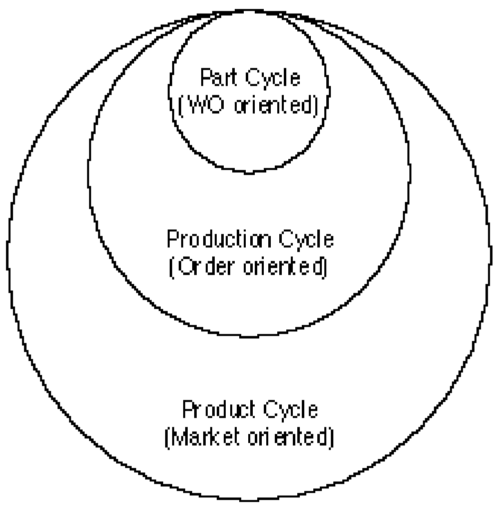


Figure 3. Enterprise Integration

These principles are utilized to develop production rules integrating CIM databases, using a modeling method discussed next.

2.2. The Modeling Method: TSER

The major requirements for a manufacturing modeling method are the ability to 1) facilitate both top-down design and bottom-up consolidation, 2) preserve and implement all the pertinent information contained in the high-level models including the process-oriented operational knowledge and data-oriented user's view in the design and control of the resultant information system, 3) support complex information structures such as CAD and CAM object classification hierarchies and decision rules for interactions among systems, and 4) automate some tasks of data and knowledge modeling [9]. The Two-Stage Entity-Relationship (TSER) model [17] along with an attendant methodology (Figure 4) was developed towards this goal.

It entails two levels of modeling constructs devised respectively for semantics-oriented abstractions (i.e., the functional constructs) and cardinality-oriented (normalized) representations (i.e., the structural constructs) of data and production rules. The constructs allow for top-down system development, as well as bottom-up design - i.e., reverse engineering of existing applications or software packages into the TSER constructs (Figure 4). The model encompasses both data and process orientations, and further combines them with a unified data and knowledge representation method. There are rigorous TSER algorithms which map from semantic to structural models and these algorithms ensure that the resulting structures are minimally in third normal form. TSER algorithms also integrate views, thus allow systematic consolidation of any number of data models. The integrity constraints built into the TSER constructs are used to facilitate the management and control of the databases.

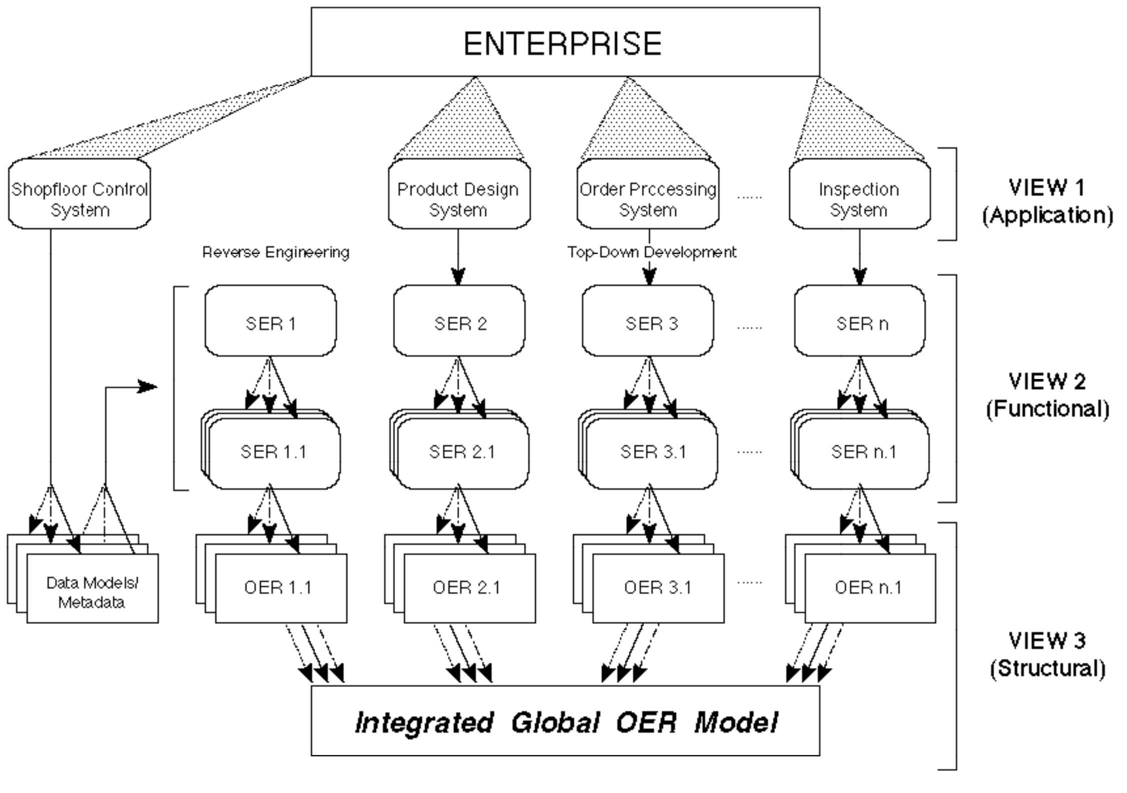
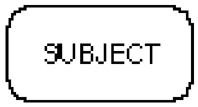
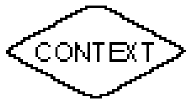


Figure 4. Enterprise Views

The Modeling Constructs :

A. Functional (Semantic) Constructs: Functional Constructs are user-oriented semantic-level constructs for object-hierarchy and process representation; used for systems analysis and information requirements modeling; referred to in TSER as the Functional (or SER for short) Level Modeling Constructs as shown in Table 1; and used exclusively to capture semantics.

Table 1. TSER Functional Model Constructs

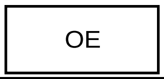
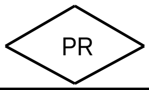
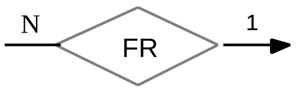
CONSTRUCT	DEFINITION AND DESCRIPTION
	Contains data items (attributes), functional dependencies among data items, *intra*-SUBJECT rules (triggers and dynamic definitions of data items belonging to a single SUBJECT), and class hierarchy (generalizes and aggregates SUBJECTs).
	Contains *inter*-SUBJECT rules (characterized by references to data items belonging to multiple SUBJECTs), typically captures directions of flows for logic (decision and control) and data (communication, etc.).

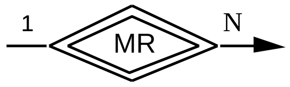
Notes :

- The full contents (as applicable) must be specified for all SUBJECTs at the leaf level of the SUBJECT hierarchy. The class hierarchy implies integrity rules for applications, but its presence is not required.
- Rules are constructed in the form of (a subset of) predicate logic where all clauses must only consist of the logical operators and the data items that have been declared or defined in the SUBJECTs. A data item may be defined as variables or parameters to represent an executable routine, algorithm, or mathematical expression.

B. Structural (Normalized) Constructs: Used as a neutral, normalized representation of data semantics and production rules from functional model for logical database design; and referred to in TSER as the structural (or OER) model. There are four basic constructs described in Table 2.

Table 2. TSER Structural Model Constructs

CONSTRUCT	DEFINITION AND DESCRIPTION
Operational Entity 	Entities identified by a singular primary key and (optional) alternative keys and non-prime attributes.
Plural Relationship 	Association of entities characterized by a composite primary key and signifying a many-to-many and independent association
Functional Relationship 	A many-to-one association that signifies characteristic or inheritance relationships. FRs represent the referential integrity constraint implied by the existence of foreign keys. The arrow side is called the *determined* and points to either an OE or a PR, while the other side is called the *determinant* and is also linked to either an OE or a PR. The primary key of the *determined* side is included as a non-prime attribute (i.e., a foreign key) of the *determinant* side.

Mandatory Relationship 	A one-to-many *fixed* association of OEs. MRs represent the existence-dependency constraint, and are symbolized as a double diamond with direction. The "1" side is linked to the *owner* OE while the arrow side points to the *owned* OE.
Notes: • In both top-down design and reverse engineering, the structural model is typically *derived* automatically from the functional model by using the TSER normalization and mapping algorithms. • While there usually are multiple functional models representing different views or application systems of an enterprise model, there always exists only one integrated structural model for the global system.	

C. The Mapping Algorithms: Mapping algorithms are used to map the functional models into structural models and to link both types with their respective metadata representations. These algorithms that are reported in [5,9,17] generate and decompose views, produce relations or other data structures, and determine integrity constraints for schema design. They also include procedures for putting all of these models (shown in Figure 4) together and into a repository called **metadatabase** and creating the metadatabase. The syntax of rules and the integrated representation of rules with data are fully discussed in [18].

We might mention that procedural knowledge can be joined with other semantic rules and asserted for the SUBJECTs. Such knowledge may represent manufacturing processes and system operating rules that are not otherwise captured. The database and knowledge structures resulted from mapping can be readily implemented in a wide range of commercially available systems including PDES EXPRESS (object-oriented) and Oracle (relational) [17].

3. The Development Of The Information Model

The reference model is instantiated into a core information model based on a CIM prototype which in turn is constructed according to core functions identified by industry. While data classes are specified, the defining effort and challenge of the instantiation is really the elaboration of contextual knowledge in the form of production rules, as discussed above in section 2.1. In the usual case where the models of more than one application system are being developed (as in the CIM prototype), a three-step process for basic global information modeling is used (see Figure 4). First, a functional model for each application system is created; second, the model is mapped to its corresponding structural model; and third, the several structural models are consolidated into a single global structural model using dependency-theoretical principles (e.g., normalization). The complete functional model is a hierarchy of models derived either from the successive decomposition of higher level SUBJECTs into submodels or according to inheritance hierarchies. By using a leveled approach, each submodel is relatively easy for users to understand and thus establishes a common reference among functional managers and information system analysts. We turn first to the key task: contextual knowledge.

3.1. Knowledge Engineering: Basic Control Rules for the Parallel Paradigm

As described above, the contextual knowledge consists of (1) control rules for information flow and (2) semantic rules/constraints for data management. The first might also be referred to as operating rules (Appendix A) and the latter integrity constraints (Appendix B). Both are corroborated with the information contents of decision processes they apply to (Appendix C). Creation of the knowledge model begins with the overall description of the enterprise tasks and their calibration with data items; i.e., inter-SUBJECT and intra-SUBJECT tasks. An intra-SUBJECT rule defines the production rules for business operations pertaining to data items of the SUBJECT, plus the selection criteria and sequence of execution for the rules linking to its sub-SUBJECTs. The business routines are executed according to the flow of actions expressed in a control rule set and operate under the influence of constraint rules. The knowledge representing the flow of control among the SUBJECTs is defined in a CONTEXT as inter-SUBJECT rules. This control rule set defines what actions are required upon a change of status of each activity in a process, and practically gives rise to the working definition of a business process involving multiple SUBJECTs.

We now develop some control rules for tracking customer orders in the CIM system as an example illustrating the knowledge engineering process used to develop the core CIM model. First, the parallel paradigm of integration is employed to identify the basic interactions needed among order processing functions, process plan system, shop floor control functions, and others. The knowledge classes defined in [7] satisfies this need. Consequently, inter-SUBJECT contextual knowledge is determined or acquired from the theory which prescribes, among other things, that a process revision is called for per certain new orders and that the status of an order in the shop floor control system needs to be sent to order processing system either at a change of status or at a request initiated from the latter. Second, status and other variables and parameters are defined in the context of the knowledge representation method of TSER and its rule language [18]; as follows:

1) **User-defined Routine :** A user-defined routine is the description of the primitive level task to be performed by a SUBJECT; it provides the input parameters and constraints, input variables, return value type, and transfer function.
2) **Status Variable :** Status variables are defined to represent the current state of each activity of the sub-systems. At process completion or interruption, a status variable is updated either by the process itself or by the status checking routines. Such change in status variables triggers the execution of business processes by initiating the processing of a set of operating rules and controls the information flows among the SUBJECTs.

All the routines that need to be considered under the purview of the CIM enterprise are defined globally at the enterprise level of the functional model, encapsulated within SUBJECTs and could be shared by different APPLICATIONs/SUBJECTs across the enterprise. They are referred to as enterprise routines to differentiate from routines that local systems might use exclusively. This architecture ensures a separation of data, function (enterprise routines), and knowledge (operating rules), thus making it possible to revise operating knowledge without rebuilding models or systems; e.g., if the data structures change, only the associated routines need to be modified, without having to update nor change the existing application.

Figure 5 shows some of the inter-SUBJECT knowledge for order processing and process plan systems; a more complete listing of rules is given in Appendix A. The order processing system provides facilities to track the order status of a specific item anywhere in the CIM system; in the meanwhile, other application systems also update their status variables that trigger the intra-SUBJECT rules. With the status variables and the checking routines, an inter-SUBJECT rule controls the sequence of actions such as downloading manufacturing data automatically to appropriate application systems based on modeled interactions, executing an application program or routines, and updating physical data in application databases.

System provided routines

```
1) every time (month, day, year, hour, minute)           /* Defines a time trigger as a time pattern to be matched */
2) changes in oepr(application, oe/pr table, detect update, detect delete, detect insert,time pattern,
                   month, day, year, hour, minute)        /* Defines a time trigger as time frequency */
```

Rule #0010 /* Inter-Subject Operating Rule between OPS <-> PPS */

```
IF (changes_in_oepr("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,TRUE,FALSE,-1,1,-1,-1,
    -1) AND                      /* If ORDER_ITEM table is updated or inserted : Check every day */

   (A.OI_STATUS = "DESIGNED") AND                 /*  if the order item status is  "DESIGNED" */

   (NOT exists (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) = B.PARTID<-(PARTREV),
    A.ORDER_LINE_ID))))                                  /*  if no such process plan  exists*/

THEN A.PLANREV<-(PLANREV) := new_revision(A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID));
                               /* copy order inf.. to Process Plan  and request new plan */

A.OI_STATUS := "IN ENG" ;                                /* change status  order item*/

A.ROUTING := 0 ;  A.PLANSTATUS := "WORKING" ;
```

Figure 5. System provided routines and inter-SUBJECT operating rule between Order Processing System and Process Plan System.

3.2. Basic Data Management Rules

The functional modeling is followed by a normalization process using a mapping algorithm to generate two normalized structural models (data and rules) that refine the representation of data and knowledge captured in the functional model. Four types of integrity constraints for data management are implied in the structural model, corresponding respectively to the four types of TSER constructs (Table 2):

1) Entity: No key component of the relation can have null value and no row in a relation may duplicate a value in the key.

2) Plural-relationship: For any record to be stored in a plural-relationship relation, there must exist records in all the relations that take part in that relationship.

3) Functional-relationship: Any record stored in the determinant relation of a functional relationship has a corresponding record in the determined relation.

4) Mandatory-relationship: For any record to be stored in a mandatory-relationship relation, there are corresponding records in the owner relation and the owned relation. Furthermore, for a record to be stored in the owned relation, there must be a corresponding record in the owner relation.

These constructs are automatically elaborated for each instance in both data and rule submodels. Thus, not only databases can be managed readily using these rules, but rules of CIM operations can also be managed in a rulebase fashion [18].

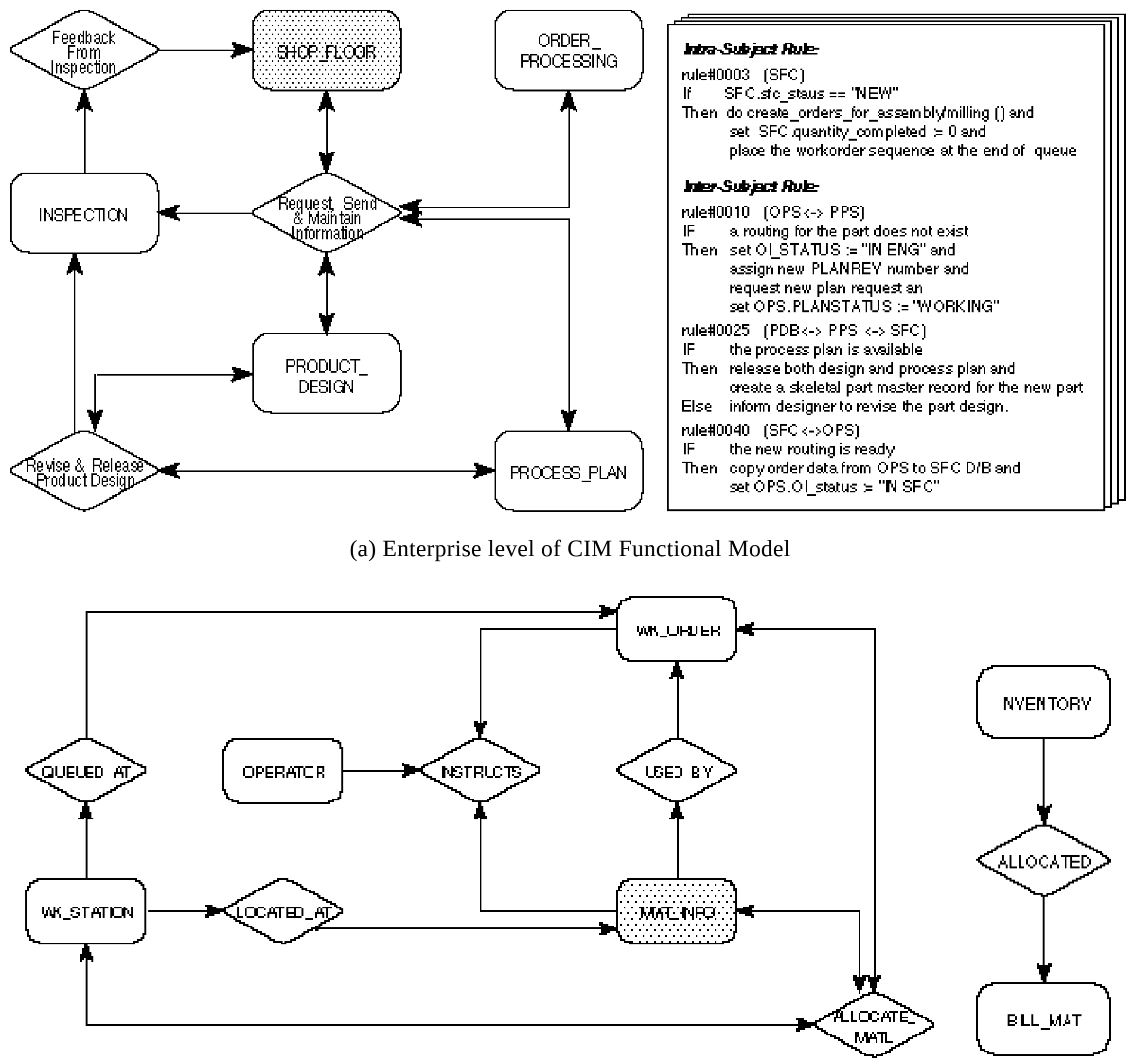
3.3. The Computer Integrated Manufacturing Core Information Model

The above core information model is completed with empirical investigations based on industrial scenarios tested at the Design and Manufacturing Institute (DMI) Computer Integrated Manufacturing (CIM) research facility at Rensselaer Polytechnic Institute (by representatives from Alcoa, Digital Equipment Corporation, GE, GM, and IBM). Table 3 presents the five component systems that were determined. These systems then drive the final instantiation of the reference model, resulting in the core model.

Table 3. Rensselaer's CIM prototype

SUB-SYSTEMS	SOFTWARE ENVIRONMENT	HARDWARE ENVIRONMENT
Order Process System (OPS)	ORACLE DBMS (Relational)	IBM RISC 6000 (UNIX)
Product Design Data Base (PDB)	ROSE (Object-Oriented)	IBM RISC 6000 (UNIX)
Process Plan System (PPS)	PC ORACLE (Relational)	IBM PC
Shop Floor Control System (SFC)	DBASE III PLUS (Relational)	IBM PC
Inspection System	ROSE (Object-Oriented)	IBM RISC 6000 (UNIX)

The enterprise view of the CIM applications, their interrelationships, data items, and functional dependencies are presented in Figure 6(a)-(c) and Appendix C. Figure 6(a) is the highest-level functional model of the CIM system and has been decomposed into functional submodels that is given in Figure 6(b)-(c) and Figure C.1. Each subsystem was modeled as a SUBJECT (rounded rectangle) with sub-system interactions as CONTEXTs (rounded diamonds). In the figures, the arrows emanating from/to the CONTEXTs indicate the flows of information and control. The production rules shown in Appendix A are encapsulated into CONTEXTs, while the data contents of some SUBJECTs are documented in Appendix B. For simplicity, only the enterprise level operating rules are shown in Figure 6(a).



```
Intra-Subject Rule:
rule#0003   (SFC)
If      SFC.sfc_staus == "NEW"
Then  do create_orders_for_assembly/milling () and
        set  SFC.quantity_completed := 0 and
        place the workorder sequence at the end of  queue

Inter-Subject Rule:
rule#0010   (OPS<-> PPS)
IF      a routing for the part does not exist
Then  set OI_STATUS := "IN ENG" and
        assign new PLANREV number and
        request new plan request an
        set OPS.PLANSTATUS := "WORKING"
rule#0025   (PDB<-> PPS <-> SFC)
IF      the process plan is available
Then  release both design and process plan and
        create a skeletal part master record for the new part
Else   inform designer to revise the part design.
rule#0040   (SFC <->OPS)
IF      the new routing is ready
Then  copy order data from OPS to SFC D/B and
        set OPS.OI_status := "IN SFC"
```

(a) Enterprise level of CIM Functional Model

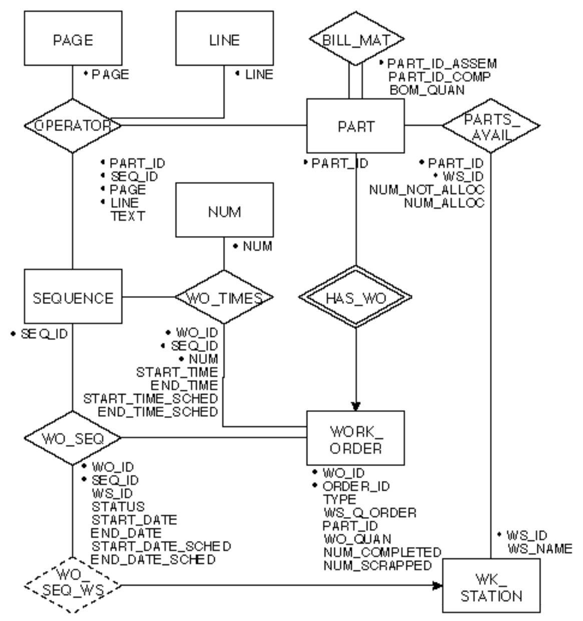
(b) Shop Floor Control System (c) MAT_INFO

Figure 6. Functional Model of Shop Floor Control System

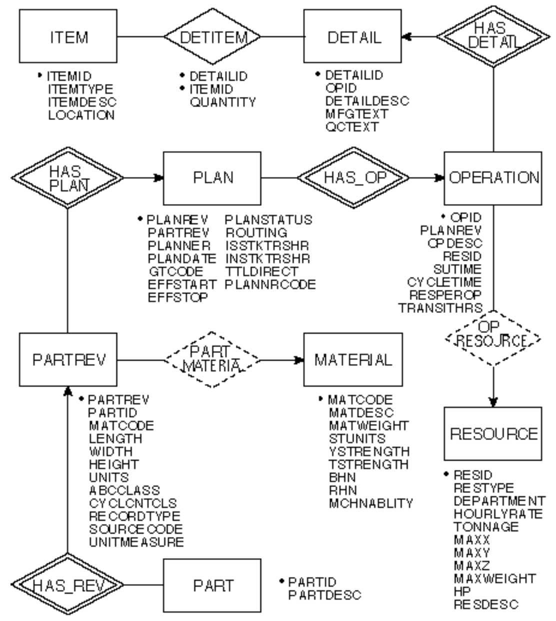
Applying TSER normalization algorithms to the data items and their functional dependencies in the shop floor control system, the process planning system, the product design system and the order processing system SUBJECTs shown in Figure 6(a) and Appendix C generates the normalized structural submodels (respectively, Figure 7(a)-(d)), and the resultant integrity constraints (functional and mandatory relation) are shown in Appendix B. Finally, Figure 8 shows the global integrated structural model resulted from the consolidation of the five CIM application submodels.

4. Application Of The Core Model for Particular Enterprises

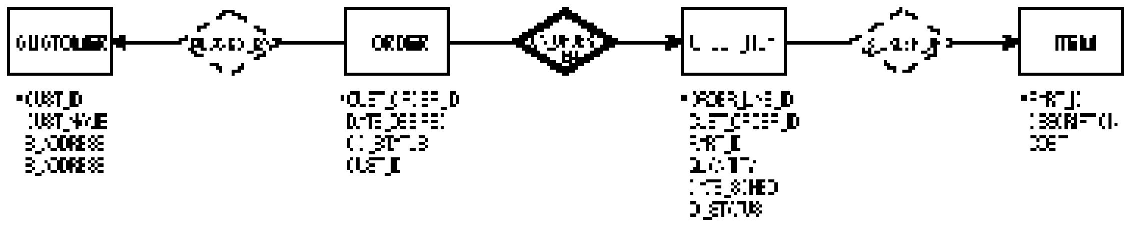
The core CIM model per se can be employed to provide a basic CIM information system with fundamental integration rules designed into it. It can also be developed with a metadatabase added on to the basic system to provide advanced capabilities such as systems growth and evolution. We discuss these progressive steps below.



(a) Shop Floor Control System

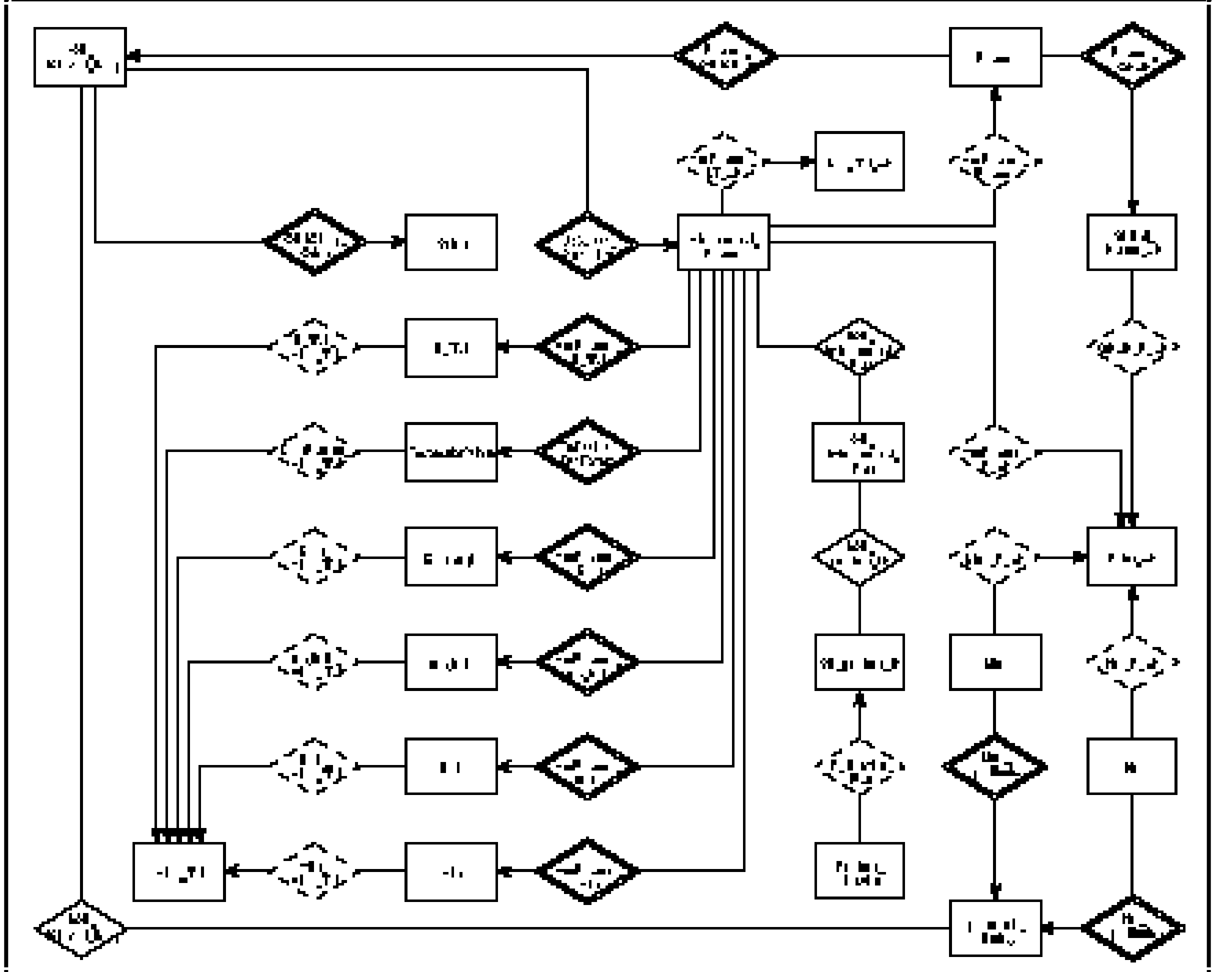


(b) Process Plan System



(c) Order Processing System

Figure 7. Structural Models of SFC, PPS, PDB, and OPS



(d) Product Design Database System

Figure 7. Structural Models of SFC, PPS, PDB, and OPS (continued)

4.1. The Procedure of Systems Development Using the Core Information Model

The structural models presented above can be immediately implemented using commonly available database management systems (DBMS), either relational or object-oriented. The object-oriented DBMS technology is less mature commercially than the relational, and its implementation tends to require more custom refinements. Thus, regarding implementation into object-oriented systems, we content ourselves with a general rule stating that the (hierarchy of) SUBJECTs from the functional models will be linked with Entities and Relationships from the structural models and thereby form a class hierarchy, where Entity-Relationship constructs constitute the leaf level objects.   One specific example of such an approach is provided in [17]

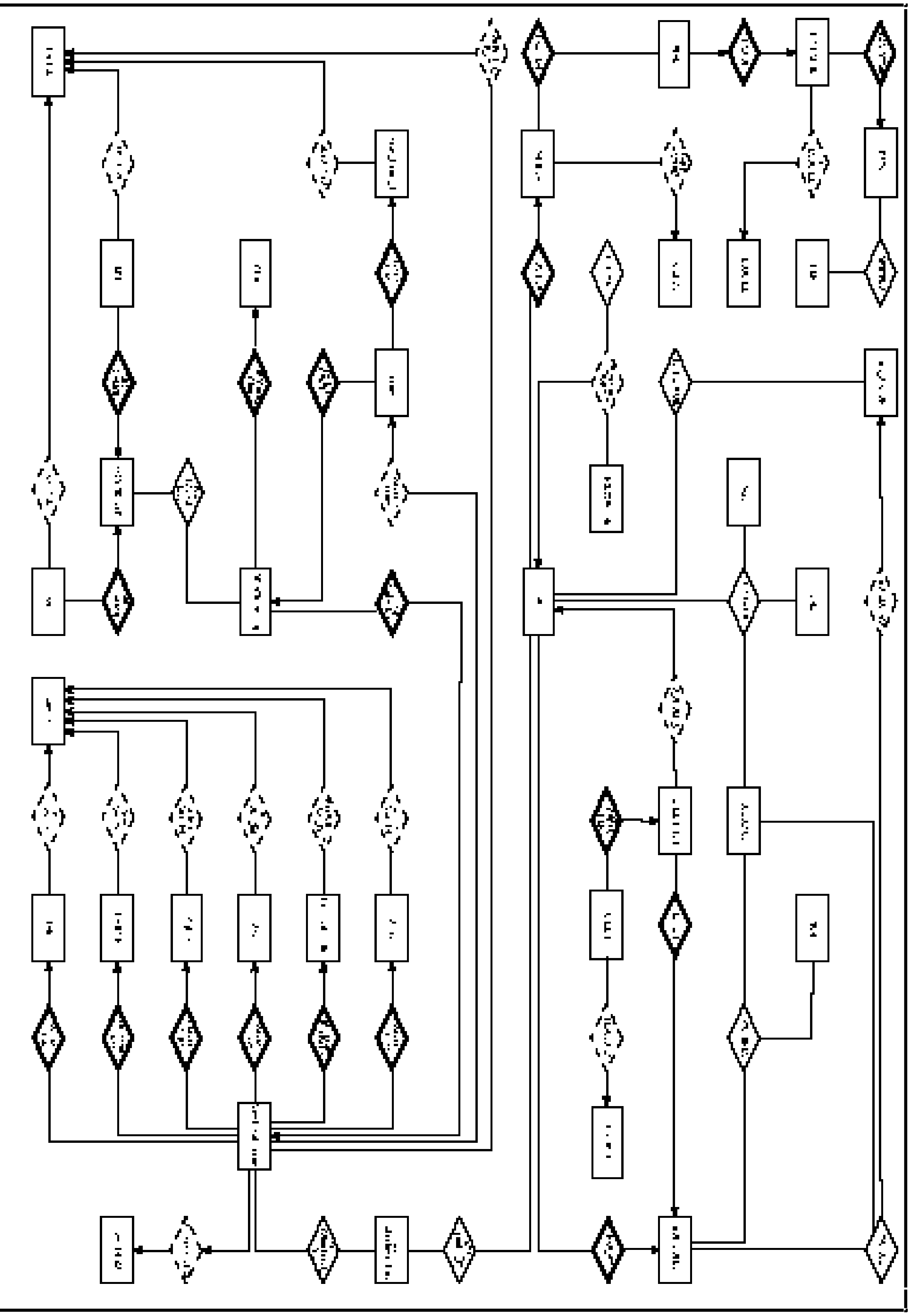


Figure 8. Integrated Structural Diagram of CIM Enterprise.

for a product database in the PDES/EXPRESS environment. For practicality, we discuss the procedure of implementation under the assumption that the manufacturer will develop their CIM information systems with off-the-shelf relational DBMS's; which seems to be the case in actuality, anyway.

The procedure is summarized below:

(a) Each of the structural models in Figure 7 maps directly into a relational schema for the particular application system, where ENTITY and PLURAL RELATIONSHIP become base relations with exactly the same primary keys and attributes. An example for Shop Floor Control is provided in Table 4. At the user's discretion, this starting design can be fine tuned in the usual manner. At this point, a sound information environment encompassing basic CIM systems for integration, but not yet integrated automatically, is in place. Each system will operate in a standalone manner and the integrating information flows must be effected through managerial measures.

Table 4. Base relation of the Shop Floor Control System

Relation Name	Definition : (Key_item(s), attribute[1],...attribute[n])
WK_STATION	(WS_ID, WS_NAME)
WORK_ORDER	(WO_ID, NUM_COMPLETED, NUM_SCRAPPEDORDER_ID, TYPE, WS_Q_ORDER, PART_ID, WO_QUAN)
WO_SEQ	(WO_ID, SEQ_ID, WS_ID, STATUS, START_DATE, END_DATE)
BILL_MAT	(PART_ID_ASSEM, PART_ID_COMP, BOM_QUAN)
PARTS_AVAIL	(PART_ID, WS_ID, NUM_NOT_ALLOC, NUM_ALLOC)
WO_TIMES	(WO_ID, SEQ_ID, NUM, START_TIME, END_TIME)
OPERATOR	(PART_ID, SEQ_ID, PAGE, LINE, TEXT)

(b) The integration knowledge - including both operating rules and data management rules - is implemented at each local system using its own programming environments. For instance, the tracking of orders will be coded as application programs or database shells at the process planning system, the shop floor control system and the order processing system, as well as any other systems involved in the operating rules. This, of course, promises to be no small undertaking; but should not be overwhelming, either, especially when compared to the integration efforts where no such core model is provided. The knowledge can be selected and implemented in an incremental manner to further alleviate the effort required. A basic system of integration is obtained, where the integrating knowledge is not globally managed and the individual systems are not supposed to be changed, added, or deleted. Nonetheless, such a basic integration is precisely most manufacturers (large and small) would be more than happy to acquire for the present.

(c) A metadatabase is created to provide further integration modeling capabilities when needed, such as that the core model will be expanded to incorporate new systems. In addition to models in Figure 7, the integrated model in Figure 8 will also be created and stored in the metadatabase, along with the functional models in Figure 6. Thus, a global data and knowledge integration is achieved at the model level by virtue of the metadatabase. The same modeling process discussed before will be able to perform the task of developing models for new systems and integrating them with existing environments (see Figure 4). The implementation of the new model will follow the same procedure here, plus necessary modifications to the code implementing rules, according to the newly consolidated contextual knowledge. An inroad into the hard task of integration management is achieved with the inclusion of the metadatabase, although the contextual knowledge and the integration architecture/configuration are not managed automatically, yet.

(d) The metadatabase is optionally augmented with an integration technology called Rule-Oriented Programming Environment (ROPE) technology [9,19] to automatically distribute the contextual knowledge from the metadatabase containing the global CIM model to local application systems and implement them into rule-based shells at individual systems. Albeit not completely automated, most of the implementation tasks described in (b) and (c) above for rules are carried out by ROPE. At this final stage, the complete integration is accomplished in an on-line and automated manner.

The above procedure itself is incremental; the user can opt to implement only the first one, as well as proceeding to any of the next three levels. Next, we discuss some implementation techniques in detail; the first three steps are combined in the discussion.

4.2. Implementation of a Basic System with a Metadatabase

Corresponding to step (c) in the above procedure, the metadatabase serves mainly as a standalone global information resources dictionary system. It contains the global catalog of information resources across the enterprise, their functional interrelationships, the transformation between information models, and the production knowledge and the interactions among the subsystems. Thus, the metadatabase may be queried of information models and control knowledge for use in further modeling or information resources management.

Both the metadatabase and the local application models can be implemented into some target (commercial) DBMS using the same techniques. For illustration, consider the structural model in Figure 7(a), using a relational DBMS. The detailed implementation of the database scheme can be described as a two-step process: 1) implementing the base relations, and 2) implementing the integrity constraints and operating rules. When the chosen DBMS explicitly supports the notion of Integrity Constraints (primary key and foreign key definition) and Database Triggers (e.g., ORACLE Version 7), the base relations and integrity rules can be implemented in a straightforward way using built-in data definition languages (DDL) as shown in Figure 9. The *pk_partrev* constraint in Figure 9 identifies the *partrev* column as the primary key of the *partrev* table, and ensures that no two part-revision codes in the table have the same part-revision id and no partrev is NULL. The operating rules can also be implemented with a non-declarative approach using Database Triggers, which are automatically fired when triggering INSERT, UPDATE, or DELETE statements.

```
CREATE TABLE partrev
  ( partrev      CHAR (5)     CONSTRAINT  pk_partrev  PRIMARY KEY,
    partid       CHAR (10),
    matcode      CHAR (15),
    length       NUMBER,
    width        NUMBER,
    height       NUMBER,
  ...........
    units        CHAR (10),
    sourcecode   CHAR (2),
    unitmeasure  CHAR (2) );
```

Figure 9. DDL statement defining a primary key constraint (Oracle v.7)

For systems that do not fully support integrity rules, a document file can be produced which details data management rules and operating rules. Next, a software tool (software development library) can be either adopted (from the field) or developed to *automatically* generated a set of procedures for the documented rules and operationalize the contextual knowledge. Such a callable library of procedures can easily be generated as standard embedded SQL for the C language and is triggered by the database application or the user whenever transactions of *insert*, *delete*, and *update* which affect the data-items are involved.

It is worth noting that referential integrities across different systems cannot be enforced by a single DBMS, but rather must have the global rules distributed and localized into these systems. Same global monitoring capabilities are also needed to maintain the consistency of rules. This is addressed by ROPE, as discussed next.

4.3. Advanced System: Growth And Change

The metadatabase can be utilized as a knowledge-based systems integrator through ROPE. This software technology calls for a new “rule” section to overlay on top of the usual data typing for efficient execution, management and maintenance of distributed knowledge models; which are implemented as rule-based shells for the component applications. Empowered by these shells, each local application can function autonomously and yet be able to coordinate information interchange without the need for global serialization (a known limitation of existing approaches to CIM integration using distributed database management systems). The metadatabase acts as the data and knowledge server that distributes operating knowledge to the local shells at the construction time and manages them during the operation time **when** there are additions or changes to the global knowledge. In the evolution process, a requirement change is reflected as a model change that is operationalized as new rule generation or rule update to the CIM application shells. Figure 10 shows the architecture of the manufacturing enterprise applications using the ROPE technology [5,9,10,19].

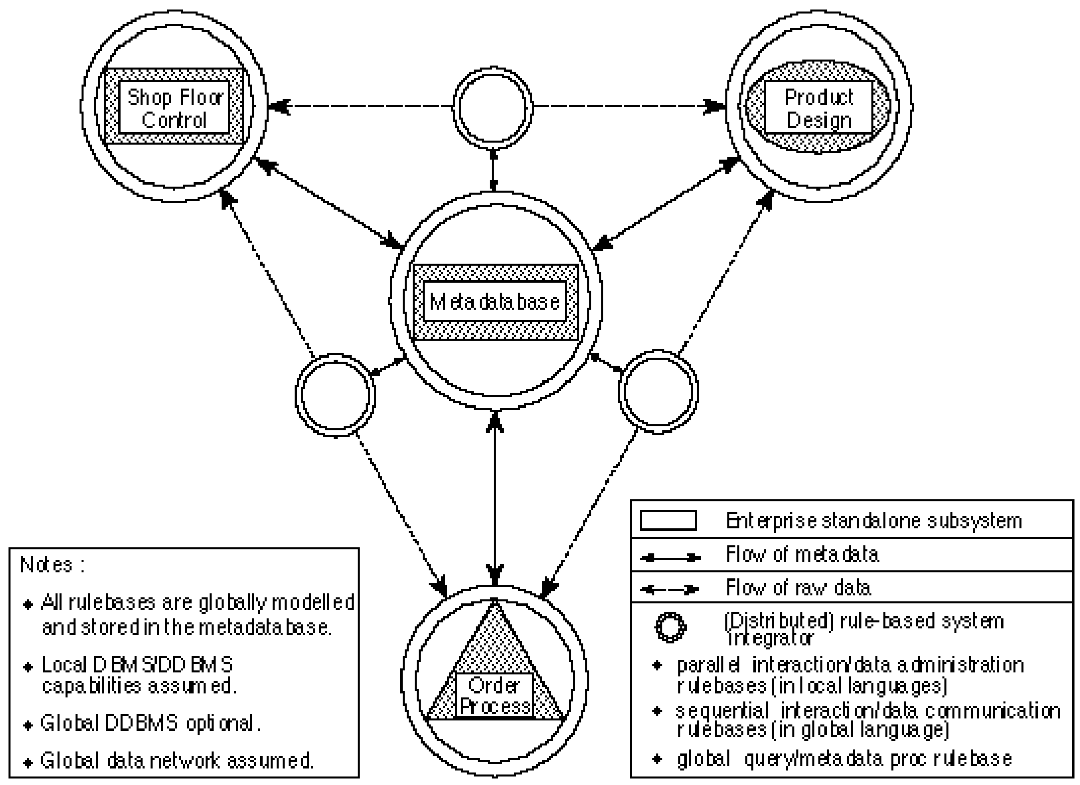

Shop Floor Control
Product Design
Metadatabase
Order Process
Notes :
All rulebases are globally modelled and stored in the metadatabase.
Local DBMS/DDBMS capabilities assumed.
Global DDBMS optional.
Global data network assumed.
Enterprise standalone subsystem
Flow of metadata
Flow of raw data
(Distributed) rule-based system integrator
parallel interaction/data administration rulebases (in local languages)
sequential interaction/data communication rulebases (in global language)
global query/metadata proc rulebase

Figure 10. Distributed Metadatabase System

4.4. Evaluation Criteria

How good is the core model and what is its performance in actuality? There are, as usual, two aspects to the answer; namely, internal validity and external validity. The first is concerned with the consistency of the model. We content that the entire derivation of the core model - from the need of such a model to the development of contextual knowledge, and all the way to the modeling methodology and the final results - is shown consistent in the previous sections. The only thing that needs to be added here is the basic assumption of the model: it is intended for CIM information systems (using databases) that does not involve directly the real time integration of physical processes such as robotic workstations. However, an extension of the metadatabase model using a new real time ROPE technology is shown to be able to integrate just such real time processes (see [20] for details). Thus, the core model allows a direct extension to include real time control rules when so desired. This compatibility is consistent with the nature of the core model being a starting point of system development for particular CIM environments, and hence supports its (internal) validity.

External validity is established in a two-pronged analysis. First, methodologically, the core model and its application approach are consistent with the common practice of the enterprise integration industry - that is, the model does meet some of the common requirements and therefore facilitate reducing some of the efforts that will otherwise be required. The additional cost for using the core model is virtually nil at the first two levels discussed in Section 4.1. The next levels require incremental costs associated with the application of the metadatabase model. This cost, however, must be compared with alternatives that accomplish the same level of functionality as the metadatabase. Empirical investigation does not exist. Conceptually, the metadatabase model is the only result reported that provides a solution of model-based architecture for adaptive and scalable integration. The fact that matters here is, present approaches to evolving an existing integration architecture are invariably less than satisfactory (in essence, they all entail a disruptive re-construction of the systems in one way or another).

Second, empirically, the proposed approach has been evaluated by the industrial sponsors of the research (see Section 3.3) and a few other companies. They in general have found the approach interesting and promising. Some have actually explored the possibilities of applying the technology to their own environment [8]. In addition, a small-sized manufacturer (100~200 employees) has compared the core model with its own integration designs. It was found that the core model does encompass most of the company's data requirements (while the operating and integrity rules go beyond these efforts), and be suitable to be "fine-tuned" for particular needs. The custom integration modeling in the case has taken a team of 12 professionals working for over three months to accomplish. The core model is poised to saving at least some of the efforts for other similar cases. Finally, the CIM prototype at Rensselaer (see Section 3.3) also offers an example where the data model and over 100 operating and integrity rules are implemented. The performance ranges from a few seconds to a few minutes for global query and rule management processing.

Therefore, the evaluation criteria for applying the proposed approach to particular CIM enterprises in actuality include the following: (1) the core model should be applicable to CIM information systems above the level of shop floor physical processes - the latter would be beyond the immediate scope of the core model; (2) the core model should encompass a significant portion of the information requirements; (3) the modeling methodology should support a ready extension/fine-tuning of the core model to suit the enterprise; and (4) the application approach should entail only major tasks that would be undertaken by the enterprise anyway (e.g., information modeling). As such, the benefits will stem mainly from the savings of efforts due to the above provisions; while the costs will result from the additional efforts required with respect to the alternatives for the same level of functionality.

5. Conclusion

Enterprise integration and re-engineering is a much-talked about but not often attempted topic in many organizations large or small, because the effort required could be prohibitive in terms of time, expertise, and all other forms of cost. This is why major players such as large consulting firms tend to use their proprietary reference models of information systems as a common starting point for systems development to many integration problems. This is also why industrial standards community has called for a variety of public reference models to facilitate integration for all companies. Yet, these efforts still tend to be tilted towards comprehensiveness and hence erred on the over-kill side. Why should, for example, a small or medium-sized manufacturer who just wants to have a decent CIM information system to assist with mundane factory-level tasks go overboard to adopt the CIM-OSA architecture (Figure 1) which extends well beyond the basic scope of requirements? Furthermore, the CIM-OSA model (or other similar models for that matter) does not provide specific contents for the manufacturer’s information model. A compact information model that satisfies the basic needs of integration for core manufacturing functions is the main contribution of this paper (sections 3, and 4). In addition, the design approach (focusing on contextual knowledge based on a theoretical reference model) itself is also a contribution (sections 1 and 2).

One key challenge this effort responded to is how to develop sound contextual knowledge utilizing and controlling data resources to accomplish integration. Data requirements and the management of databases in a basic CIM facility are relatively well-understood and can be satisfied by a number of commonly available database technologies. Knowledge is altogether another story. Control knowledge is traditionally treated separately from databases and routinely excluded from data modeling efforts. Yet, it is precisely the integration of data and knowledge in a unified representation that is needed for the core information model; and this integration cannot be considered on an ad hoc piece-meal basis. This is where the integration theory and the modeling method (section 2) came in and made a contribution to the final development of the compact reference model this paper presents.

The core model is based on a CIM prototype developed according to industrial scenarios by the joint effort of Alcoa, DEC, GE, GM, and IBM at Rensselaer. Both the data sub-model and knowledge sub-model are fully discussed (section 3 and Appendices A, B and C). Their representation and possible extension or customization for particular companies are accomplished by using the TSER methodology (section 2 and 3). To illustrate how such a core information model could be employed for actual development, an implementation design is devised. The scenario is simply that the model would be directly adopted for real use in a CIM system; whereby using commercial packages is shown to be sufficient to develop and operate the total environment (section 4).

This line of effort - namely, reference models - is both important and promising. Much more work is evidently needed. More theoretical principles and empirical investigations are critical to developing better reference models. More specialization would be the key to effecting what CIM-OSA termed respectively “particular” and “partial” models for specific industries and firms. Finally more integration technologies are required to automate modeling tasks as well as improve the processing performance, both of which are still bottlenecks in integration environments.

REFERENCES

[1] Hsu, C. and C. Skevington, "Integration of Data and Knowledge in Manufacturing Enterprises: A Conceptual Framework," *Journal of Manufacturing Systems*, Vol. 6, No. 4, 1987 pp. 277-285.

[2] Sage, A. P., "Systems Engineering and Information Technology: Catalysts for Total Quality in Industry and Education," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 22, No. 5, 1992 pp. 883-864.

[3] ESPRIT Consortium AMICE (eds.), "*Open System Architecture for CIM,*" *Springer*-Verlag, 1989.

[4] SofTech, "Integrated Computer-Aided Manufacturing (ICAM)," *Architecture PART III/Volume VI Composite Information Model of 'Manufacture Product' (MFG1),* AFWAL-TR-82-4063 Vol. VI, Wright-Patterson AFB, OH, 45433, Sept. 1983.

[5] Hsu, C. and L. Rattner, "Information Modeling for Computerized Manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 20, No. 4, 1990, pp. 758-776.

[6] Hsu, C. and L. Rattner, "Integration in Manufacturing: An Information Theoretic Perspective," *Production and Operations Management,* 1(3), 1992, pp. 286-294.

[7] Rattner, L., "Information Requirements for Integrated Manufacturing Planning and Control: A Theoretical Model," Unpublished Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1990.

[8] Hsu, C. and A. Rubenstein, "Enterprise Information Management for Global Manufacturers," *4th International Conference on Computer Integrated Manufacturing and Automation Technology,* October 1994 (forthcoming).

[9] Hsu, C., G. Babin, M. Bouziane, W. Cheung, L. Rattner, and L. Yee, "Metadatabase Modeling for Enterprise Information Integration," *Journal of Systems Integration,* Vol. 2, No. 1, 1992, pp. 5-39.

[10] Hsu, C., L. Gerhardt, D. Spooner, and A. Rubenstein, "Adaptive Integrated Manufacturing Enterprises: New Information Technology for the Next Century," *IEEE Transactions on Systems, Man, and Cybernetics (forthcoming).*

[11] Yeh, R.T., "Notes on Concurrent Engineering," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 4, No. 5, 1992, pp. 407-414.

[12] Zeidner, L. and Y. Hazony, "Seamless Design-to-Manufacture," *Journal of Manufacturing Systems,* Vol. 11, No. 4, 1992, pp. 269-284.

[13] Shrensker, W., ed., "CIM: A Working Definition," *Society for ManufacturingEngineers,* 1990.

[14] Chryssolouris, G., M. Lee, and M. Domroese, “The Use of Neutral Networks in Determining Operational Policies for Manufacturing Systems,” *Journal of Manufacturing Systems,* Vol. 10, No. 2, 1991, pp. 166-175.

[15] Reimann, M. and J. Sarkis, “An Architecture for Integrated Automated Quality Control,” *Journal of Manufacturing Systems,* Vol. 12, No. 4, 1993, pp. 341-355.

[16] Harhalakis, G., C.P. Lin, K.Y. Moy, and H. Hillion, “A Facility-Level CIM System,” *Proceedings 1988 International CIM Conference on Computer Integrated Manufacturing*, IEEE Computer Society, 1988, pp. 253-263.

[17] Hsu, C., Y. Tao, M. Bouziane, and G. Babin, “Paradigm Translations in Integrating Manufacturing Information Using a Meta-Model,” *Information Systems Engineering*, Vol. 1, No. 3, 1993, pp. 325-352

[18] Bouziane, M. and C. Hsu, “A Rulebase Model for Integration of Data and Knowledge in Multiple system Environments,” *International Journal of Artificial Intelligence Tools* (forthcoming).

[19] Babin, G., “Adaptiveness in Information Systems Integration,” Unpublished Ph.D. Thesis, Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1993.

[20] Shaefer, O. and C. Hsu, “Distributed Rulebases for Real Time Process Control on the Shop Floor: the Metadatabase Approach,” *ASME Journal of Engineering for Industry* (forthcoming).

APPENDIX A. Operating Rules

```
IF    (changes_in_oepr ("PROCESS_PLAN","PLAN",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,2,-1) AND
      (ROUTING >0) AND
      (ORDER_PROCESSING.COST =0) AND
      (PLANSTATUS <> "RELEASED"))
THEN  PLANSTATUS := "RELEASED" ;
      ORDER_PROCESSING.COST := TTLDIRECT ;
      OI_STATUS := "ENGNRD" ;
      enter_new_plan (PROCESS_PLAN.PLANREV<-(PLANREV)) ;

      IF       ROUTING code is assigned and routing is available (greater than 0) and      /* OPS_PPS */
               ORDER_PROCESSING.COST is not assigned and
               process PLANSTATUS is not “RELEASED”
      THEN     set PLANSTATUS := "RELEASED";
               set ORDER_PROCESSING.COST := TTLDIRECT ;
               set OI_STATUS := "ENGNRD" ;
               do plan for next part.

IF    (changes_in_oepr ("PROCESS_PLAN","PLAN",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,2,-1) AND
      (ROUTING >0) AND
      (COST <>0) AND
      (PLANSTATUS <> "RELEASED"))
THEN  PLANSTATUS := "RELEASED" ;
      OI_STATUS := "ENGNRD" ;
      enter_new_plan (PROCESS_PLAN.PLANREV<-(PLANREV)) ;

      IF       ROUTING code is assigned and routing is available (greater than 0) and      /* OPS_PPS */
               ORDER_PROCESSING.COST is already assigned and
               process PLANSTATUS is not “RELEASED”
      THEN     set PLANSTATUS := "RELEASED";
               set OI_STATUS := "ENGNRD" ;
```

do plan for next part.

```
IF   (changes_in_oepr ("PROCESS_PLAN","PLAN",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,2,-1) AND
     (ROUTING = -1))
THEN PLANSTATUS := "HOLD" ;
     request_design_revision (PROCESS_PLAN.PARTREV<-(PARTREV)) ;
```

IF ROUTING code is assigned and routing is not available (= -1) ***/* PPS_PDB */***
THEN set PLANSTATUS := "HOLD" ;
Request design revision.

```
IF   (changes_in_oepr ("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,-
     1,30) AND
     (OI_STATUS = "ENGNRD"))
THEN enter_remote (CUST_ORDER_ID<-(ORDER_LINE_ID),ORDER_PROCESSING.PART_ID<-
     (ORDER_LINE_ID),QUANTITY) ;
     OI_STATUS := "IN SFC" ;
```

IF the new routing is ready ***/* SFC_OPS */***
THEN copy work order information from OPS D/B to SFC D/B.
set OI_STATUS := “IN SFC” ;

```
IF   (changes_in_oepr ("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,-
     1,30) AND
     (B.OI_STATUS = "NOT ASSGND") AND
     (exists(B.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) = A.PROCESS_PLAN.PARTID<-
     (PARTREV),B.ORDER_LINE_ID)))
THEN B.OI_STATUS := "IN SFC" ;
     enter_remote (B.CUST_ORDER_ID<-(ORDER_LINE_ID),B.ORDER_PROCESSING.PART_ID<-
     (ORDER_LINE_ID),B.ORDER_PROCESSING.QUANTITY) ;
```

IF OI_STATUS = “NOT ASSGND” and */* SFC_OPS_PPS */*
process plan for the part already exists
THEN set OI_STATUS := “IN SFC” and
copy order information from OPS D/B to SFC D/B and

```
IF   (changes_in_oepr ("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,FALSE,FALSE,-1,-1,-1,-
     1,1) AND
     (WO_QUAN <> ORDER_PROCESSING.QUANTITY))
THEN WO_QUAN := ORDER_PROCESSING.QUANTITY ;
```

IF there is any update in ORDER_ITEM table and ***/* SFC_OPS */***
work order quantity changes.
THEN copy new work order quantity to WO_QUAN in SFC **;**

```
IF   (every_time (-1,-1,-1,1,0) AND
     (is_assembly (ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) AND
     (SHOP_FLOOR.STATUS = "ENDED")))
THEN ORDER_PROCESSING.OI_STATUS := "ASSEMBLED" ;
```

IF the part is in assembly process and ***/* SFC_OPS */***
SHOP_FLOOR.STATUS is "ENDED"
THEN set ORDER_PROCESSING.OI_STATUS := "ASSEMBLED" ;

```
IF   (every_time (-1,-1,-1,1,0) AND
     (is_assembly (ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) AND
     (SHOP_FLOOR.STATUS = "START")))
THEN ORDER_PROCESSING.OI_STATUS := "ASSEMBLY" ;
```

IF the part is in assembly process and ***/* SFC_OPS */***
SHOP_FLOOR.STATUS = "START"
THEN set ORDER_PROCESSING.OI_STATUS := "ASSEMBLY" ;

```
IF   (every_time (-1,-1,-1,1,0) AND
     (NOT is_assembly (ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) AND
     (SHOP_FLOOR.STATUS = "ENDED")))
THEN ORDER_PROCESSING.OI_STATUS := "MILLED" ;
```

IF the part is not in assembly process and ***/* SFC_OPS */***
SHOP_FLOOR.STATUS = "ENDED"
THEN set ORDER_PROCESSING.OI_STATUS := "MILLED" ;

```
IF   (every_time (-1,-1,-1,1,0) AND
     (NOT is_assembly (ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) AND
     (SHOP_FLOOR.STATUS = "START")))
THEN ORDER_PROCESSING.OI_STATUS := "MILLING" ;
```

IF the part is not in assembly process and ***/* SFC_OPS */***
SHOP_FLOOR.STATUS = "START"
THEN set ORDER_PROCESSING.OI_STATUS := "MILLING" ;

```
IF    (every_time (-1,-1,-1,0,10) AND
      (NUM_SCRAPPED > (0.01 * WO_QUAN)))
THEN  SHOP_FLOOR.STATUS := "HOLD" ;
      move_to_end_of_queue (WO_ID<-(WO_ID SEQ_ID)) ;
      notify_operator (WO_ID<-(WO_ID SEQ_ID)) ;
```

```
      IF      number of scrapped item is greater than 1% of work order        /* SFC_INSPECT ION*/
      THEN    set SHOP_FLOOR.STATUS := "HOLD" ;
              move it to end of queue;
              notify  operator
```

```
IF    (changes_in_oepr ("ORDER_PROCESSING","ORDER",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,12,-1)
      AND
      ((OD_STATUS = "DONE") AND
      (max (END_DATE) > DATE_DESIRED)))
THEN  request_process_revision (PART_ID<-(WO_ID)) ;
```

```
      IF      OD_STATUS = "DONE" and                                          /* OPS_SFC_PPS */
              any of order is past_due
      THEN    request process planner for future revision.
```

```
IF    (changes_in_oepr ("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,6,-
      1) AND
      (A.OI_STATUS = "NOT ASSGND") AND
      (NOT exists (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) = B.product_ID<-
      (product_ID),A.ORDER_LINE_ID))))
THEN  A.OI_STATUS := "IN DESIGN" ;
      prepare_new_design_request (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) ;
      A.$Product_Version<-($Product_Version) := new_revision (A.ORDER_PROCESSING.PART_ID<-
      (ORDER_LINE_ID)) ;
```

```
      IF      OI_STATUS is not assigned and                                   /* OPS_PDB */
              there is no existing design for the PARTID
      THEN    set OI_STATUS “IN DESIGN” and
              prepare new design request (Copy order information in OPS to PDB) and
              assign new Product_Version number
```

```
IF    (changes_in_oepr ("ORDER_PROCESSING","ORDER_ITEM",TRUE,FALSE,TRUE,FALSE,-1,-1,-1,6,-
      1) AND
      (A.OI_STATUS = "NOT ASSGND") AND
      (exists (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) = B.product_ID<-
      (product_ID),A.ORDER_LINE_ID))))
THEN  A.OI_STATUS := "DESIGNED" ;
      prepare_new_Pro_plan_request (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID)) ;
```

```
      IF      OI_STATUS is not assigned and                                   /* OPS_PDB_PPS */
              there exists a design for the PARTID
      THEN    set OI_STATUS “DESIGNED” and
              request new process planning.
```

```
IF    (changes_in_oepr ("PRODUCT_DESIGN","Product_Version",TRUE,FALSE,FALSE,FALSE ,-1,-1,-
      1,-1,1) AND
      (exists (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) =
      B.INSPECTION.PART_ID,A.ORDER_LINE_ID))))
THEN  A.OI_STATUS = "DESIGNED" AND
      A.INSPECTION.STATUS<-(PART) := "ASSGND" ;
```

```
      IF      the design is updated or added and                              /* INSPECTION_OPS_PDB */
              inspection plan for the part exists
      THEN    set OI_STATUS “DESIGNED” and
              set INSPECTION.STATUS = "READY"
```

```
IF    (changes_in_oepr ("PRODUCT_DESIGN","Product_Version",TRUE,FALSE,FALSE,FALSE ,-1,-1,-
      1,-1,1) AND
      (NOT exists (A.ORDER_PROCESSING.PART_ID<-(ORDER_LINE_ID) =
      B.INSPECTION.PART_ID,ORDER_LINE_ID))))
THEN  A.OI_STATUS = "DESIGNED" AND
      convert_design_model_to_inspection_model (A.Product_Version<-
      (Product_Version),A.product_ID<-(Product)) ;
      Inspection_plan_request (A.Product_Version<-(Product_Version),A.product_ID<-
      (Product)) ;
```

```
      IF      the design is updated or added and                              /* INSPECTION_OPS_PDB */
              inspection plan for the part not exists
      THEN    set OI_STATUS “DESIGNED” and
              convert design data to inspection data and
              request new Inspection plan
```

```
IF    (changes_in_oepr ("PRODUCT_DESIGN","INSP_PART",TRUE,FALSE,FALSE,FALSE ,-1,-1,-1,-1,5)
      AND
      inspection.status = “COMPLETE”)
```

```
THEN  record_ins_result (Pard_ID,Serial#,Actual_tol).
```

IF inspection is done **/* INSPECTION_SFC */**
THEN record inspection results into PDB

APPENDIX B. Integrity Constraints

Table B.1. Functional Relationship

Appl.	Functional Relationship	Determinant	Determined	Determinant Data Item(s)	Determined Data Item(s)
OPS	IS_ORDER_OF	ORDER_ITEM	ITEM	ORDER_LINE_ID	PART_ID
	PLACED_BY	ORDER	CUSTOMER	CUST_ORDER_ID	CUST_ID
SFC	WO_SEQ_WS	WO_SEQ	WK_STATION	WO_ID, SEQ_ID	WS_ID
PPS	PART_MATERIAL	PARTREV	MATERIAL	PARTREV	MATCODE
	OP_RESOURCE	OPERATION	RESOURCE	OPID	RESID
PDB	SmFeature_Feature	Sheetmetal_Feature	Feature	$Sheetmetal_Feature	feature_ID
	GD2D_P_2D	Global_Datum_2D	Point_2D	$Global_Datum_2D	$Point_2D
	SmFeature_P_2D	Sheetmetal_Feature	Point_2D	$Sheetmetal_Feature	$Point_2D
	Arc_P_2D	Arc	Point_2D	$Arc	$Point_2D
	SmFeature_LT_2D	Sheetmetal_Feature	Loc_Tol_2D	$Sheetmetal_Feature	$Loc_Tol_2D
	Rect_Size_Tol	Rectangle	Size_Tol	$Rectangle	$Size_Tol
	Line_P_2D	Line	Point_2D	$Line	$Point_2D
	U_Tab_Size_Tol	U_Tab	Size_Tol	$U_Tab	$Size_Tol
	RectPattern_Size_Tol	RectangularPattern	Size_Tol	$RectangularPattern	$Size_Tol
	Keyhole_Size_Tol	Keyhole	Size_Tol	$Keyhole	$Size_Tol
	Hole_Size_Tol	Hole	Size_Tol	$Hole	$Size_Tol
	Slot_Size_Tol	Slot	Size_Tol	$Slot	$Size_Tol
	P_Version_Part	Product_Version	OE_product_ID	$Product_Version	product_ID

Table B.2. Mandatory Relationship

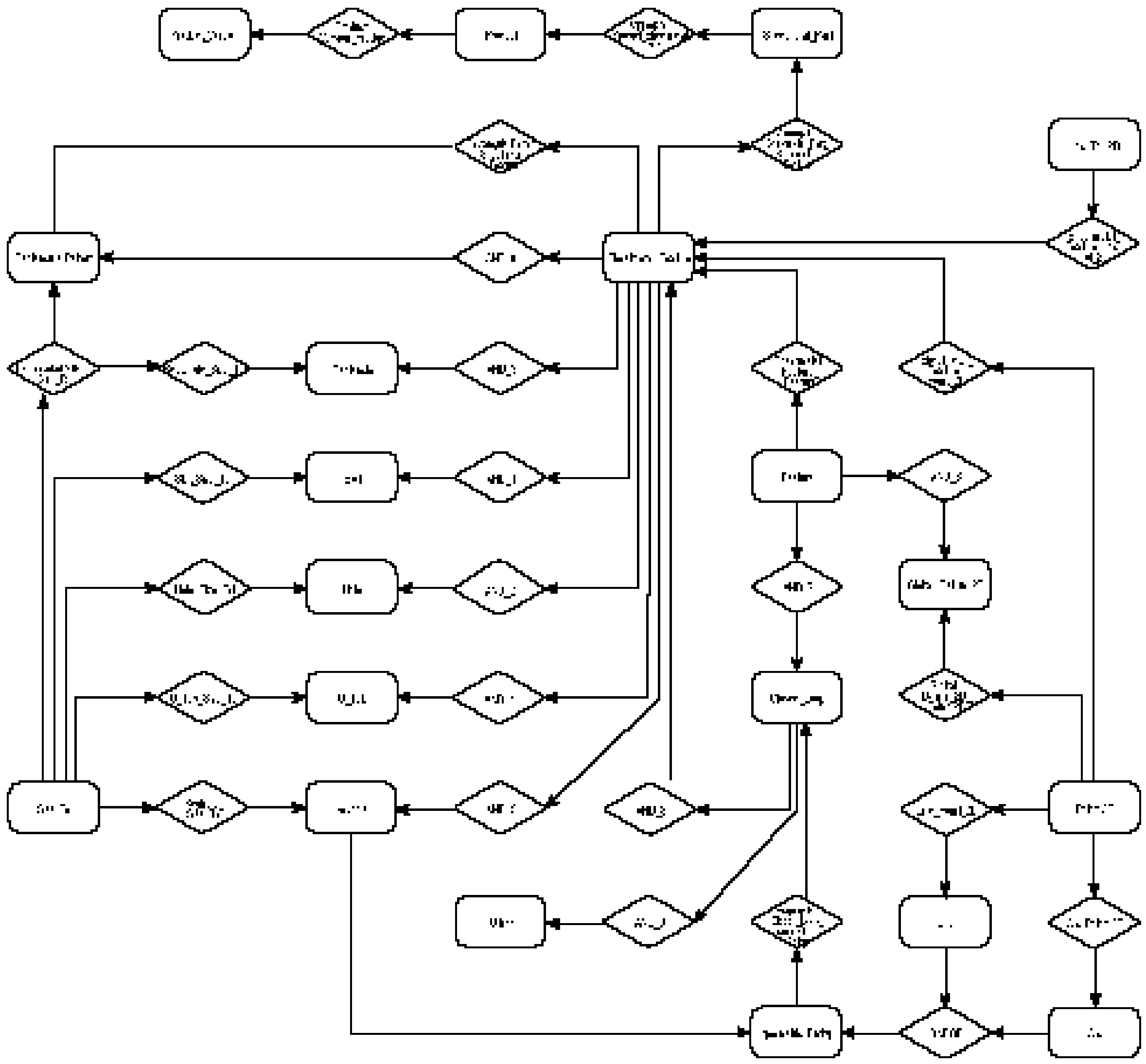
Appl.	Mandatory Relationship	Owner	Owned	Owner Data Item(s)	Owned Data Item(s)
OPS	ON_ORDER_ITEM	ORDER	ORDER_ITEM	CUST_ORDER_ID	ORDER_LINE_ID
SFC	HAS_WO	PART	WORK_ORDER	PART_ID	WO_ID
PPS	HAS_REV	PART	PARTREV	PART_ID	PARTREV
	HAS_PLAN	PARTREV	PLAN	PARTREV	PLANREV
	HAS_OP	PLAN	OPERATION	PLANREV	OPID
	HAS_DETAIL	OPERATION	DETAIL	OPID	DETAILID
PDB	Feature_OE$Cloop	Feature	OE_$Closed_Loop	feature_ID	$Closed_Loop
	Feature_GD2D	Feature	Global_Datum_2D	feature_ID	$Global_Datum_2D
	OE$Cloop_SmFeature	OE_$Closed_Loop	Sheetmetal_Feature	$Closed_Loop	$Sheetmetal_Feature
	OE$Cloop_Other	OE_$Closed_Loop	Other	$Closed_Loop	other_ID
	SmFeature_U_Tab	Sheetmetal_Feature	U_Tab	$Sheetmetal_Feature	$U_Tab
	SmFeature_Rect	Sheetmetal_Feature	Rectangle	$Sheetmetal_Feature	$Rectangle
	SmFeature_RectPattern	Sheetmetal_Feature	RectangularPattern	$Sheetmetal_Feature	$RectangularPattern
	SmFeature_Keyhole	Sheetmetal_Feature	Keyhole	$Sheetmetal_Feature	$Keyhole
	SmFeature_Hole	Sheetmetal_Feature	Hole	$Sheetmetal_Feature	$Hole
	SmFeature_Slot	Sheetmetal_Feature	Slot	$Sheetmetal_Feature	$Slot
	Line_geoEntity	Line	geometric_Entity	$Line	$geometric_Entity
	Arc_geoEntity	Arc	geometric_Entity	$Arc	$geometric_Entity

APPENDIX C. Functional Model

Table C.1. Data Items and Functional Dependencies

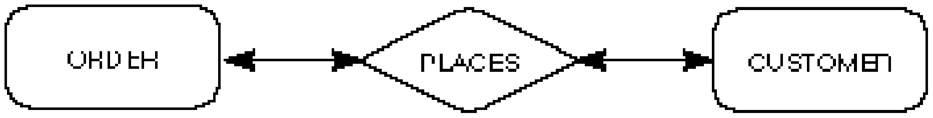
Appl.	SUBJECT	FDs
OPS	CUSTOMER	(CUST_ID) -> (CUST_NAME, B_ADDRESS, S_ADDRESS)
	ORDER	(CUST_ORDER_ID) --> (OD_STATUS, CUST_ID, DATE_DESIRED) (PART_ID) --> (COST, DESCRIPTION) (CUST_LINE_ID) --> (CUST_ORDER_ID, PART_ID, QUANTITY, OI_STATUS, DATE_SCHED)
SFC	MAT_INFO	No FDs. Super-subject encompassing subjects BILL_MAT and INVENTORY
	BILL_MAT	(PART_ID_ASSEM, PART_ID_COMP) --> (BOM_QUAN)
	INVENTORY	(PART_ID_COMP, PART_ID, WS_ID) --> (NUM_NOT_ALLOC, NUM_ALLOC)
	WK_STATION	(WS_ID) --> (WS_NAME)
	OPERATOR	(PART_ID, SEQ_ID, PAGE, LINE) --> (TEXT)
	WK_ORDER	(WO_ID) --> (WO_QUAN, TYPE, NUM_COMPLETED, ORDER_ID, NUM_SCRAPPED, PART_ID, WS_Q_ORDER) (WO_ID, SEQ_ID) --> (STATUS, END_DATE, WS_ID, START_DATE_SCHED, END_DATE_SCHED) (WO_ID, SEQ_ID, NUM) --> (END_TIME, START_TIME, END_TIME_SCHED, START_TIME_SCHED)
PPS	MATERIAL	(MATCODE) --> (YSTRENGTH, TSTRENGTH, RHN, BHN, MATDESC, MATWEIGHT, STUNIT, MCHABLITY)
	ITEM	(ITEMID) --> (ITEMTYPE, ITEMDSC, LOCATION)
	RESOURCE	(RESID) --> (TONNAGE, MAXX, MAXZ, MAXY, HP, RESTYPE, DEPARTMENT, HOURLYRATE, MAXWEIGHT, RESDESC)
	PART	(PARTID) --> (PARTDESC) (PARTREV) --> (SOURCECODE, ABCCLASS, HEIGHT, CYCLCNTCLS, WIDTH, PARTID, UNITMEASUR, LENGTH, MATCODE, UNITS, RECORDTYPE)
	PLAN	(PARTREV) --> (PARTID) (PLANREV) --> (PLANNER, ROUTING, PLANSTATUS, TTLDIRECT, PLANNRCODE, EFFSTART, SSTKTRSHR, PLANDATE, PARTREV, GTCODE, INSTKTRSHR)
	OPERATION	(OPID) --> (PLANREV, RESID, CYCLETIME, RESPEROP, OPDESC, TRANSITHRS, SUTIME) (PARTREV) --> (PARTID) (PLANREV) --> (PARTREV)
	DETAIL	(DETAILID) --> (MFGTEXT, OPID, DETAILDESC) (OPID) --> (PLANREV) (PARTREV) --> (PARTID) (PLANREV) --> (PARTREV)
	DETITEMS	(DETAILID) --> (OPID) (DETAILID, ITEMID) --> (QUANTITY) (OPID) --> (PLANREV) (PARTREV) --> (PARTID) (PLANREV) --> (PARTREV)
PDB	Feature	(feature_ID) --> (feature_ID)
	Point_2D	($Point_2D) --> (x, y)
	Line	($Line) --> (start_Point, end_Point)
	Arc	($Arc) --> (center_Point, start_Point, end_Point)
	geometric_Entity	($geometric_Entity) --> ($Line, $Arc)
	Closed_Loop	($geometric_Entity, $Closed_Loop) --> (geo_List$U$L) ($Closed_Loop) --> (feature_ID)
	Other	(other_ID) --> ($Closed_Loop)
	Global_Datum_2D	($Global_Datum_2D) --> (feature_ID, origin, x_Datum)
	Sheetmetal_Feature	($Sheetmetal_Feature) --> ($Closed_Loop, centerpoint, theta_Rot, datum_Feature, location_Tol, gT_Position_Flag, gT_Position_Tol)
	Loc_Tol_2D	($Loc_Tol_2D) --> (x_Tol_Plus, x_Tol_Minus, y_Tol_Plus, y_Tol_Minus)
	Size_Tol	($Size_Tol) --> (plus minus)
	Hole	($Hole) --> ($Sheetmetal_Feature, diameter, diameter_Tol, gT_Circularity_Flag, gT_Circularity_Tol)

Slot	($Slot) --> ($Sheetmetal_Feature, s_length, s_width, size_Tolerance, gT_Profile_Flag, gT_Profile_Tol)
Rectangle	($Rectangle) --> ($Sheetmetal_Feature, height, width, height_Tol, width_Tol, corner_Radius, corner_Radius_Tol, corner_Radius_Flag)
U_Tab	($U_Tab) --> ($Sheetmetal_Feature, height, width, height_Tol, width_Tol, gap, gap_Tol, corner_Radius_Flag, corner_Radius, corner_Radius_Tol)
Keyhole	($Keyhole) --> ($Sheetmetal_Feature, circleDiameter, slotDiameter, k_length, circle_Tol, slot_Tol, length_Tol, corner_Radius_Flag, corner_Radius,corner_Radius_Tol)
RectangularPattern	($RectangularPattern) --> ($Sheetmetal_Feature, no_OF_Columns, no_OF_Rows, x_Pitch, y_Pitch, stagerFlag, pitch_Tol, patternType)
Sheetmetal_Part	($Sheetmetal_Feature, $Sheetmetal_Part) --> (plate_Charact$L) ($Sheetmetal_Part) --> ($Sheetmetal_Part)
Product	($Sheetmetal_Part, product_ID) --> (components$L) (product_ID) --> (product_ID)
Product_Version	($Product_Version) --> (product)

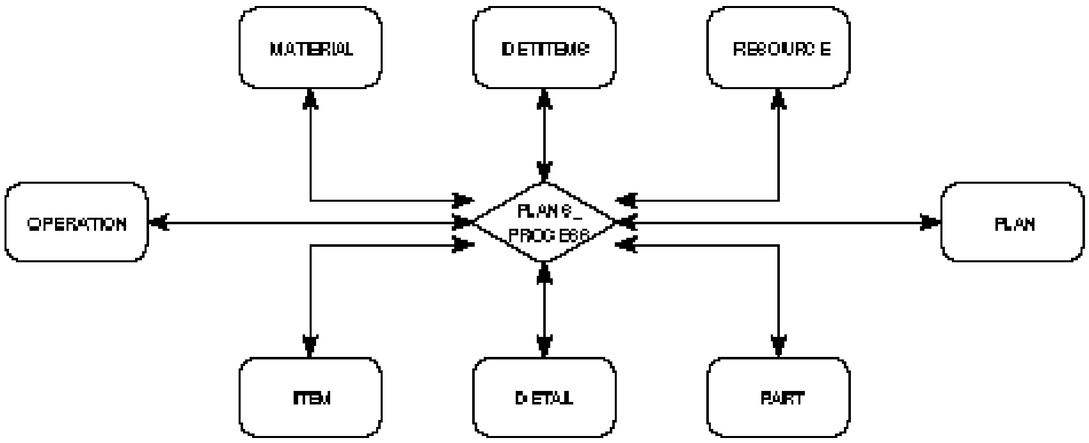


(a) Product Design System (PDB)

Figure C. Functional Model of PDB ,OPS, PPS



(b) Order Processing System (OPS)



(c) Process Plan System (PPS)

Figure C. Functional Model of PDB ,OPS, PPS (continued)