# Paradigm Translations in Integrating Manufacturing Information Using a Meta-model : The TSER Approach

**Cheng Hsu* — Yi-Cheng Tao* — M'hamed Bouziane** — Gilbert Babin***

** Rensselaer Polytechnic Institute*
*Decision Sciences and Engineering Systems,*
*Troy, New York, USA*
*12180-3590*

*** Digital Equipment Corporation*
*Nashua, New Hampshire, USA*
*03062.*

*RÉSUMÉ. L'intégration de l'information nécessaire pour la manufacturation requiert la traduction de systèmes fonctionels variés, utilisant divers paradigmes (par exemple entité-relation, relationel et objet). Afin de faciliter cette tâche potentiellement acca–blante, un meta-modèle servant de schéma conceptuel à l'intégration autant lors du développement que lors de l'exécution est nécessaire. Le présent article présente une structure spécifique de ce concept et discute du potentiel du modèle TSER (Two-Stage Entity-Relationship) en tant que meta-modèle pour les méthodes orientés-objet, le modèle relationel et les modèles entité-relation. Les connections au modèle PDES/EXPRESSsont inclues.*

*ABSTRACT. Manufacturing information integration requires translations among various functional systems' information models which are based on different paradigms (e.g., Entity-Relationship, Relational, and Object). In order to facilitate this potentially overwhelming task, a meta-model is needed to serve as the conceptual schema for models integration at both development phase and run-time. This paper presents a particular framework for the concept and discusses the promises of the TSER (Two-Stage Entity-Relationship) model as a meta-model for Object-Oriented Methods, the Relational model, and Entity-Relationship models. The connections to the PDES/EXPRESS model are included.*

*MOTS CLÉS: Méta-modèles, Intègration de l'entreprise, Modélisation de l'information, Gestion des données.*

*KEY WORDS: Meta-Models, Enterprise Integration, Information Modeling, Data Management.*

## 1. Meta-Model : The Conceptual Schema for Integrated Multi-Model Environments

Business databases, manufacturing databases, and engineering design databases have traditionally followed different paths of evolution and espoused different paradigms, although they are all based on the same information technology. To compound the situation further, each paradigm has also prompted a number of different modeling tools since the advent of Computer-Aided Software Engineering (CASE). The integration of manufacturing functions must deal with the full scope of paradigm translation problem facing business, manufacturing and engineering design databases pertaining to the enterprise. Current strategies commonly employed by companies range from the formation of the so-called strategic alliances to reduce the number of different systems involved, all the way to the adoption of a common standard for all systems. The problem with a common standard is, ironically, the lack of a standard that is acceptable to the industry, suitable to all requirements, and adaptable to changing technology. The strategic alliance approach tends to rely on "hand-shaking", i.e., developing peer-to-peer solutions on an ad hoc basis, and hence is prone to suffering from the sheer size of (potential) number of pairs and is vulnerable to change to the contents of the models as well as to the compositions of the alliance.

Therefore, the notion of using a meta-model to anchor these paradigm translations has emerged recently. It can be traced back to the early IRDS work in U.S. [ANS 85, DOL 87] and the CIMOSA project in EC [ESP 89], among others. Major ongoing industry-led efforts under ISO include PDES community’s project on developing a general information model for product design and process planning , and the IRDS community’s endeavor under ANSI for formulating an encompassing framework involving common data and knowledge representation methods and ontology [FUL 92, PAT 92, SOW 91]. Considering their all-encompassing nature, these visions may turn out to more likely be high-level reference models than particular meta-model solutions that can be immediately implemented for the needing manufacturing enterprises. In any case, many manufacturers can use compact solutions that are non-overwhelming in effort, reasonable in cost, and yet sufficiently support their integration needs. A particular meta-model system using the Two-Stage Entity-Relationship (TSER) method [HSU 85, HSU 92] and the Metadatabase model [HSU 87, HSU 90, HSU 91, HSU 92, HSU 93b] is developed to provide a compact solution for this purpose.

The substantiation of the meta-model concept in this system spans three levels:

(1) At the modeling (or metadata) level, the meta-model is a neutral paradigm serving as the common representation method that all paradigms are translated into.

(2) At the models integration (or metadata management) level, the meta-model is a generic metadata schema abstracting and structuring all models into an integrated enterprise metadatabase.

(3) At the information management (or data instances) level, the meta-model is the integrated enterprise model contained in the metadatabase.

Clearly, the meta-model system is envisioned to not only support CASE tools management and paradigm translation per se, but also utilize the resultant metadata capabilities to directly facilitate the management of application information systems across the enterprise. The above concept is illustrated in Figure 1, where the Paradigm Translation Knowledge Kernel contains mapping knowledge for a layered mapping approach progressing from particular tools to their generic paradigms, then to the common core, as depicted in Figure 2.

The basic advantages of this meta-model approach are that, the complexity of the translation problem is simplified from exponential (pairwise connection) to polynomial through the use of such an enterprise conceptual schema. The impact of any change to existing application models and systems, or addition of new ones, on the rest of the enterprise is also minimized. The feasibility of the system is demonstrated in a prototype at the Adaptive Integrated Manufacturing Enterprises Program at Rensselaer Polytechnic Institute, Troy, New York, U.S.A. [HSU 93a, HSU].

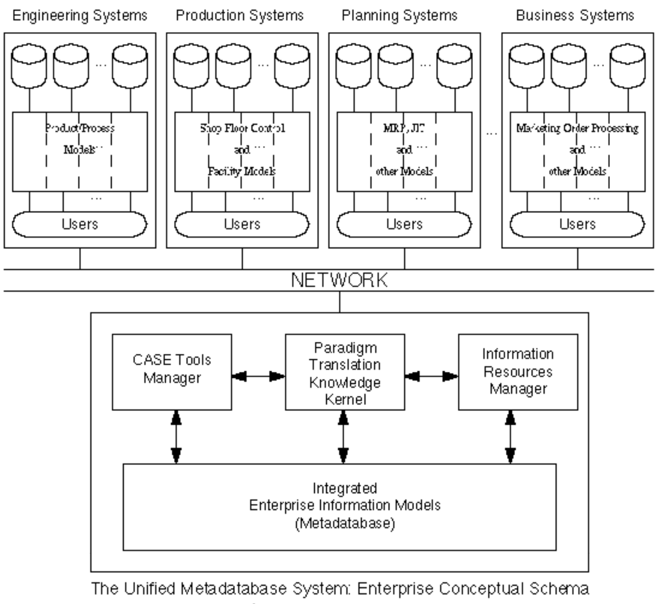

Engineering Systems
Production Systems
Planning Systems
Business Systems
Product/Process Models
Shop Floor Control and Facility Models
MRP, JIT and other Models
Marketing Order Processing and other Models
Users
Users
Users
Users
NETWORK
CASE Tools Manager
Paradigm Translation Knowledge Kernel
Information Resources Manager
Integrated Enterprise Information Models (Metadatabase)
The Unified Metadatabase System: Enterprise Conceptual Schema

**Figure 1.** *Enterprise Information Integration Using Meta-Models*

The TSER method provides the meta-model at the first, modeling level; while the Metadatabase model which is itself represented using TSER avails the second (through its schema) and the third (through its instantiation) levels of meta-model. This completeness is a unique characteristic of the meta-model: it provides a seamless integration of functionalities at all levels as well as a logical synergism of these metadata concepts. In comparison, virtually all other results in the field fall either in the category of information modeling (level one plus aspects of level two) [ANS 85, FUL 92, PAT 92, SOW 91], or in multiple-databases management (level 3 plus aspects of level 2 — see [BAB 93, BOU 91, CHE 91, SHE 90] for some analyses and surveys). The value of such an integrated solution for the enterprise information integration problems is sufficiently documented in the metadatabase literature. TSER alone, as a meta-modeling method, offers an implementation-proven compactness if not comprehensiveness in the field. As discussed above, we believe that a major contribution in compactness that facilitates the actual implementation and therefore fosters the real integration is needed.

With the particular concept of meta-models in this system formulated in this section for the first time, Section 2 reviews TSER and Metadatabase based on our previous publications (especially [HSU 91, HSU 92]). The modeling and paradigm translation algorithms are discussed respectively in Sections 3 and 4. The prototyping environment is summarized in Section 5 along with concluding remarks. Sections 1, 3, and 4 represent original contributions of this paper to meta-models.

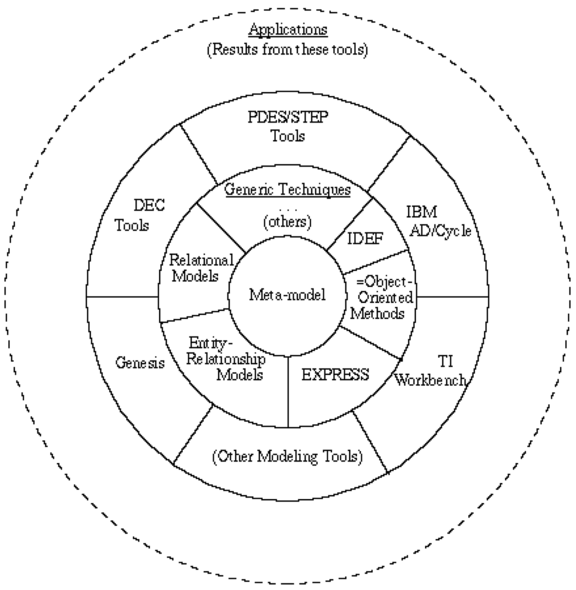


**Figure 2.** *The Paradigm Translation Knowledge Kermel for Layered Mapping*

## 2. The Two-Stage Entity-Relationship (TSER) Approach to Meta-Models

The perfect but unlikely meta-model evidently must support all conceivable paradigms now and in the future. A little more realistic expectation is that an ideal method would satisfy several requirements that can be derived from the literature and the above vision. The requirements include (1) the ability to represent major classes of knowledge concerning manufacturing processes in operating, control, and decision areas, (2) the inclusion of all pertinent models from a multi-stage analysis and design life cycle, (3) the neutral accommodation of heterogeneous data views (especially relations and object hierarchies), and (4) the unified representation of the full content of these metadata.

Concerning the representation of knowledge, a major element of these classes of knowledge is "flows" or dynamics that can be characterized with digraphs, such as data flows and control flows in the model. These global flows cannot be sufficiently represented through any implicit methods that embed knowledge into an encapsulation around data types. Such encapsulation methods are amenable mainly to localized triggers, contingency definitions, and other similar application logic. A combination of encapsulation and digraph techniques using certain generic primitives would suffice the need.

An objective for combining multi-stage models is isomorphism. That is, the mapping from a model at one stage of the life cycle to a model at the next stage should be complete and accurate. A corollary of this isomorphism is a reversible modeling process where, after the initial development of systems has long been completed, the information models that resulted from the initial analysis and design can be recovered from the metadatabase.

On the issue of data representation, a clear need is to encompass heterogeneity as exemplified by the relational approach and the object-oriented paradigm, two recent alternative models. It is evident that each has revealed certain unique but fundamental elements of the science of data modeling that have to be considered by any general model. In particular, the relational approach has, among other things, established the dependency theory for data integration (i.e., removal of redundancy) and the principle of separating applications and user views from common data structures; while the object-oriented paradigm asserts (or reasserts) the data abstraction hierarchy and integration of certain classes of transactions with data modeling.

There is probably no ideal method existing presently that satisfies all of the above requirements. However, the meta-model scope and structure aim directly at *subjects* of databases and their *contexts* for information integration, leading to a particular representation method which is discussed below.

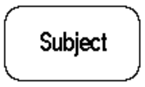
## 2.1. *The Modeling Method : TSER*

The Two-Stage Entity-Relationship (TSER) model was first developed to integrate some tasks of system analysis with database design in complex enterprises and was later expanded to include knowledge representation. It entails two levels of modeling constructs devised respectively for semantics-oriented abstractions (i.e., the functional constructs defined below) and cardinality-oriented (normalized) representations (i.e., the structural constructs defined below) 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. There are rigorous TSER algorithms which map from semantic to structural models and these algorithms ensure that the resulting structures are in at least third normal form. TSER algorithms also integrate views, thus allowing 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 metadatabase.

### 2.1.1. *The Functional (Semantic) Modeling Constructs*

The first stage features user-oriented semantic-level constructs for object-hierarchy and processes representation. They are used for system analysis and information requirements modeling; referred to in TSER as the Functional (or SER) Level Modeling Constructs; and employed exclusively for capturing semantics. The constructs include the following :

Subject:



Primitives: Contains data items (attributes), functional dependencies (among data items), *intra*-SUBJECT rules (triggers, events, and dynamic definitions of data items belonging to a single SUBJECT), and association hierarchies (explodes as well as generalizes and aggregates SUBJECTs). Specific types of association include : classification (single and multiple inheritance — mandatory relationships), grouping (partial and function-defined inheritance — functional relationship) and process (no inheritance — no integrity implications on information management).
Description: Represents functional units of information such as user views and application systems, and is analogous to frame or object.

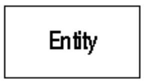
Context:

Primitives: Contains *inter*-SUBJECT rules (characterized by references to data items belonging to multiple SUBJECTs), typically includes directions of flows for logic (decision and control) and data (communication, etc.). It is allowed to contain only descriptive rules used for holding together a set of SUBJECTs or SER models that otherwise do not have direct associations.
Description: Represents interactions among SUBJECTs and control knowledge such as business rules and operating procedures and is analogous to process logic.
Note:
(1) 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.
(2) 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 (except for certain key words such as *do* and *execute*.). A data item may be defined to represent an executable routine, algorithm, or mathematical expression.
(3) The three types of association of Subjects may exist simultaneously in a model and thereby create three abstraction plains for Subjects; namely, (i) generalization and strong aggregation (the classification type), (ii) ad hoc aggregation (the grouping type), and (iii) simple decomposition similar to Data Flow Diagram and IDEF (the process type).

### 2.1.2. *The Structural (Normalized) Modeling Constructs*

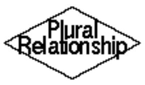
The second stage is concerned with providing a neutral normalized representation of data semantics and production rules from functional model for logical database design; and referred to in TSER as the structural (or OER) model. There are four basic constructs described below.

Operational Entity (OE):



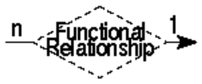
Description: Entities; identified by a singular primary key and (optional) alternative keys and non-prime attributes, and implies Entity Integrity : no component (attribute) of a primary key may accept null value, and no primary key (as a whole) may accept duplicate values.

Plural Relationship (PR):



Description: Association of entities; characterized by a composite primary key and signifying a many-to-many and independent association, and implies Associative Integrity : each of the PK's component attributes must be a PK of some Entity and its values in the PR must match that in the Entity.

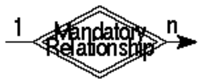
Functional Relationship (FR):



Description: A many-to-one association that signifies characteristic traits or composition relationships (corresponding to the grouping type of association of SUBJECTs). FRs represent the referential integrity constraints implied by the existence of foreign keys.

The arrow side is called the *determined* side 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. Referential Integrity : the value of every foreign key in the *determined* must either be null or be identical to some PK value in the *determinant.*

Mandatory Relationship (MR):



Description: A one-to-many *fixed* association of OEs that signifies derived and inheritance relationships (corresponding to the classification type of association of SUBJECTs). 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. Existence Dependency : when the *owner* instance is deleted, then all *owned* instances associated will it must also be deleted; and there is a foreign key implied in the *owned* whose value must match exactly the *owner*’s PK value.

Note:

(1) 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.

(2) 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.

In sum, underlying these constructs at both stages are two types of primitives: data items and predicate logic. Therefore, the basic structure of TSER metadata is characterized by (1) data representation as relations, (2) knowledge representation in the form of production rules, and (3) the two representations being tied via data items.

## 2.2. *The Meta-Model for Information Modeling*

The translation of usual paradigms into TSER would be best comprehended by first examining the inner working of TSER as a modeling paradigm itself. From the perspective of an information modeling methodology, the TSER method may be described as follows. There are actually a variety of modeling perspectives supported by TSER, including data modeling, knowledge modeling, and functional modeling that combines both. The set of constructs for functional modeling is comprised of SUBJECTs and CONTEXTs and is used to represent data semantics and knowledge as viewed from an application level or a systems analysis perspective. These constructs may be employed in their entirety or just subsets of them, depending on the perspective and tasks intended. To illustrate how SUBJECTs and CONTEXTs may be employed separately to perform some major tasks for data or knowledge modeling, they are compared to certain traditional notions in Table 1. Perform traditional data modeling tasks for, say, relational database design would not require CONTEXTs (or even the intra-subject knowledge). A model using SUBJECTs alone would suffice the requirements of semantic data modeling, and then lead to the normalized structure (OER) through the attendant mapping algorithms. Similarly, Subjects could serve as only certain reference points for recognizing and grouping rules via Contexts in rulebase modeling. When, on the other hand, both SUBJECT and CONTEXT are used, the resulting model would encompass what is usually referred to as functional models of structured systems analysis. Again, in the case of TSER, the functional level model would include data semantics and knowledge which would then be structured into normalized combined representations, i.e., the structural level model. The structural modeling constructs are one type of entity and three special types of associations that refine the representation of data and production rules captured in the functional model of logic design. They, too, may be decoupled from the functional constructs and used for modeling in their own right. In this manner, they can be compared to the traditional Entity-Relationship model except for the rigorous definitions in TSER. These definitions ensure proper data structures and integrity rules for the design of databases or rulebases or their combination.

In the usual case where the models of more than one system are being developed, a three-step process for basic global information modeling is used. First, a functional model (hierarchy) for each application system is created; second, each 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). When systems integration is actively formulated on top of this basic model, step 2 would be expanded to provide a single functional model. That is, the several functional models would be integrated into a global functional model by creating and populating *inter*-application CONTEXTs with the control knowledge and operating rules that define the interactions among application systems. This process represents the traditional top-down modeling in the common life cycle of systems analysis and design.

| Constructs | Perspective | Comparable Methods |
|---|---|---|
| Subject (used alone) | Data Modeling | Semantic Data Models and Object-Oriented Models |
| Context (used alone) | Knowledge Modeling | Process and Flow Models, and Rule-Based. Models |
| Subject and Context | Functional Modeling | Functional Models (e.g., DFD and IDEF0) |
| Entity and Relationships (used directly) | Data Modeling | Entity-Relationship Models |

**Table 1.** *The TSER Modeling Portfolio*

Paradigm translation, on the other hand, is based on bottom-up reverse engineering. This process, however, will piggy-back on the top-down capability of TSER. In essence, reverse engineering calls for a general guideline whose specificity depends largely on the particular systems to be translated in reverse engineering. The general guideline employs TSER functional constructs to represent the local models and then proceed from there following the usual (top-down) methodology. For example, base relations, views, or data files would be represented as subjects, with additional data semantics being modeled into functional dependencies, intra-subject rules, or contexts. A specific algorithm for file systems is presented in [HSU 90]. Relational systems and other Entity-Relationship models are clearly compatible with being modeled as SUBJECTs. A remaining major task is to translate Object models into TSER; this task is accomplished in Section 4. The above logic is depicted in Figure 3, which includes both top-down modeling and reverse engineering approaches.

## 2.3. *The Metadatabase Conceptual Schema Using TSER*

The Metadatabase is constructed from combining all models and views in Figure 3, along with other classes of metadata such as software and hardware resources. All of these combined metadata are organized according to a conceptual schema which is the meta-model for models (or metadata resources) integration. The populated metadatabase directly participates in the management of enterprise information and facilitates the integration of local systems. Therefore, the metadatabase becomes a data instance level meta-model not only to support CASE tools and other passive metadata applications, but also to effect functionalities needed in global query processing and systems interactions [CHE 91, HSU 92].

The development of this generic metadatabase schema is fully reported in [HSU 91]. In a nutshell, the schema follows directly from Figure 3 and contains all three classes of views (application, functional, and structural) plus resources views, in all enterprise models and the logical interrelationships among these views towards effecting data and knowledge integration and management. When fully elaborated, the structure is shown in Figure 4. This structure is based on TSER modeling constructs (a class of self-descriptiveness) and hence is generic excepting Software and Hardware Resources, which are extensible depending on particular enterprises.

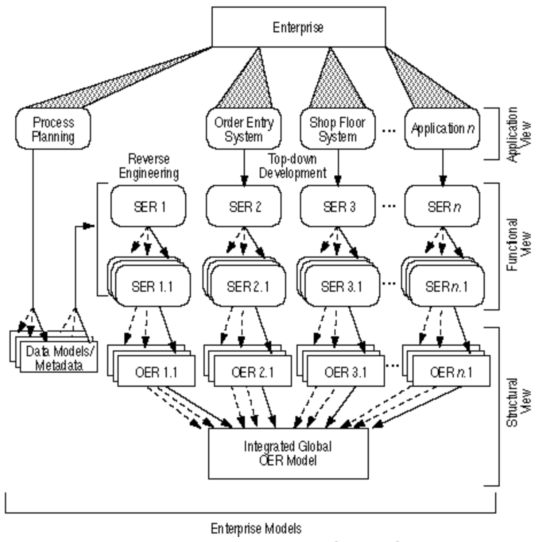


**Figure 3.** *TSER As a Meta-Model for Information Modeling*

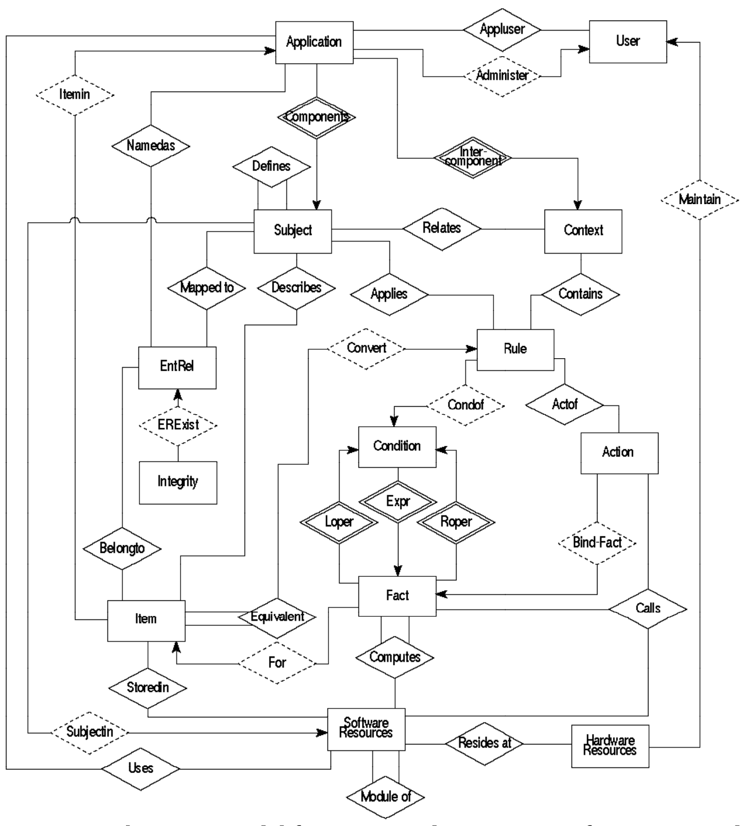


**Figure 4.** *The Meta-Model for Integrated Enterprise Information Models*

## 3. Basic Mapping Algorithms of TSER

There are two basic groups of mapping algorithms supporting TSER as a modeling method. The first derives normalized structures from the functional model and the second uses the resulting structures to design database schema, including relations and integrity rules, object-oriented hierarchies, and CODASYL data structures. In addition, the rules that are captured in the functional model (Context and Subject) are also grouped into a persistent rulebase model associated with the data structures that use them/they use. This grouping is performed directlyaccording to the schema of the rulebase; which is developed using the same principles of the metadatabase schema and the same dependency theory that the structural mapping for database design employs. In addition to these two basic groups are algorithms that perform paradigm translations for specific target models.

## 3.1. *Functional-to-Structural Mapping Algorithm*

The mapping process is carried out in three main steps. The first step, called DECOMPOSITION, creates a submodel for each SUBJECT in the hierarchy (excepting the process type of association - or, decomposition plain, where only the leaf level SUBJECTs are substantive for mapping) and analyzes its basic cardinality. The second, NORMALIZATION step, improves and simplifies the data structures within each submodel based on dependency theory. Finally the third step, CONSOLIDATION, links and merges these submodels to produce an OER model corresponding to the SER in the input. The global variables used by the three steps are categorized as follows:

A. _Fields_ : which represent the fundamental (generic or worldlike) objects.. The value of a field can be either a simple atom such as the name of an entity or relationship, or a group of atoms such as the attributes of an entity or relationship. These fields are the following:

NAME : represents the name of a structure (entity or relationship).
KEY : the primary key of a structure.
ALTKEY : the alternate keys.
NONKEY : the non-key attributes of a structure.
TYPE : A designation telling whether the structure is an OE, PR, MR, or FR.
INVOLVEDIN : A list of the entities and relationships involved in a PR, or associated with an OE.
OWNER, OWNED : used only in MR structures to denote the owner and owned side of an MR, respectively.
DETERMINANT, DETERMINED : used only with FR structures to denote the determinant and the determined sides of an FR, respectively.
FDS : a list used only in the SE's to contain the list of all the functional dependencies among its attributes.
PR-DANT : a list used to contain the list of all composite determinants in FDS which are partially involved with key attributes.

B. _Structures_ : Two fields or more are grouped together to form a structure, which can be a semantic entity or an operational entity/relationship. There are six major types of structures :

SE : represents a Subject; it has four fields : NAME, KEY, NONKEY, and FDS.
OE : represents an entity, consisting of five fields : NAME, TYPE, KEY, NONKEY, and INVOLVEDIN.
PR : represents a plural relationship. It contains the same fields as an OE.
MR : represents a mandatory relationship, consisting of the fields: NAME, TYPE, KEY, OWNER, and OWNED.
FR : represents a functional relationship using four fields : NAME, TYPE, DETERMINANT, and DETERMINED.
GROUP : this structure is used only in the Normalization step. It represents a collection of non-key attributes related to each other by a FD. It consists of two fields: DANT for the determinant side and DED or the determined side of a FD.

C. _Lists_ : Groups of structures sharing some properties are linked together to form lists. The major lists used in the mapping algorithms are:

SER : a list of all the Subjects in the input.
SM : a list of Subjects and Contexts in a submodel.
SMS : a list of all the submodels.
OER : the results of the mapping, which is a list of all the entities and relationships in the OER model.
GROUPS : a list whose elements are of type GROUP.
ASS-ATT : a temporary list, used to keep track of the non-key attributes already assigned to the OE's created during the decomposition.

### 3.1.1. *The DECOMPOSITION Step*

An SE having a composite primary key (PK) is recognized as an association of some generic entities. Thus, it is decomposed into a plural relationship (PR) and certain operational entities associated with it. The PR inherits the name and the primary key of a particular OE in the relationship. Then the non-key attributes of the SE in question are assigned among the PR and its OE's according to the functional dependencies ( and also be asserted by the semantic rules in the knowledge base). However, if the SE's PK is singular, it is identified as an OE. The result of this process in either case is declared as a submodel. This information of submodels provides an access path for information production. Procedure for this step is given below:

```
Procedure DECOMPOSITION (SE);
begin
        PK:=SE.KEY;
        AK:=SE.ALTKEY;
        NPK:=SE.NONKEY;
        FD:=SE.FDS;
        PR-DANT:=Get_PR-DANT(SE.FDS);
        if  Length(PK) > 1 then
        begin           /* case of composite key */
             PR:=create('PR');
             PR.NAME:=SE.NAME;
             PR.KEY:=PK;
             ASS-ATT:=NIL;
             Dants:=FirstSet(PR-DANT);
             while Dants ≠ NIL do
                  SubPR:=create('PR');
                  SubPR.NAME:=Naming('PR', Dants);
                  SubPR.KEY:=Dants;
                  SubPR.NONKEY:=AssignAtt(Dants, NPK, FD);
                  Insert(SubPR, SM);
                  Dants:=NextSet(Dants, PR-DANT);
             end;
             K:=First(PK);        /* start with the first component in PK */
             while K ≠ NIL do
             begin                /* for each component in the PK an OE is created */
                  OE:=Create('OE');
                  OE.NAME:=Naming('OE',K);       /* give the OE a name */
                  OE.KEY:=K;
                  OE.NONKEY:=AssignAtt(K, NPK, FD);
                  OE.INVOLVEDIN:=PR.NAME;
                  PR.INVOLVEDIN:=AddList(PR.INVOLVEDIN, OE.NAME);
                  UpdatePR_Involved(OE, PR-DANT, PK, SM);
                  Insert(OE, SM);           /* Store OE's in corresponding submodel */
                  K:=Next(K, PK);           /* proceed to the next component in PK */
             end;
             PR.NONKEY := Difference(NPK, ASS-ATT);
             Insert(PR, SM);
        end;
        else            /* case of singular PK, just one OE is created in that SM */
        begin
             OE:=Create('OE');
             OE.NAME:=SE.NAME;
             OE.KEY:=PK;
             OE.NONKEY:=NPK;
             Insert(OE, SM);
        end;
end;
```

### 3.1.2. *The NORMALIZATION Step*

After decomposition, the next step is to identify the OE's and OR's that are embedded in the data abstraction in a submodel. Therefore, the second step decomposes the structures further to ensure at least the Boyce-Codd normal form(BCNF) for all OE's and the third normal form for all OR's. It first unbundles the nested FD structures (if any) in the PR via the definition of new PR's, then removes the transitive functional dependencies (TFD) by representing them through functional relationships(FR's). At the end, it compiles all the relationships as necessary to include the semantic information.

Procedures NESTEDRELATIONS (ER) and TFD (ER) are devised for the further decomposition, but they are omitted from here due to space considerations; This decomposition is performed recursively, using two list for the same model: namely, TSM, which contains the OE's and PR's to be checked for TFD's, and SM which stores an OE or OR only if it does not contain any TFD.

All the relationships in a submodel after the normalization are compiled according to both the modeling rules and the operating knowledge (e.g., fixed associations between entities). The TSER system, when implemented, guides the user to make decisions based on the semantic rules. Examples include deciding when a PR or a FR should be declared as an MR; or providing a singular identifier for a PR; or converting an MR into an OE (see [HSU 85] for details). These rules are a part of the modeling knowledge in the knowledge base. This normalization step is summarized in the procedure below:

```
Procedure NORMALIZATION (SM);
begin
    TSM:=NIL;              /* TSM is a temporary submodel to alternate results */
    ER:=First(SM);
    while  ER ≠ NIL do          /* second normal form */
    begin
        if  ER.TYPE  =  'PR'
            then NESTEDRELATIONS(ER)
            else  Insert(ER, TSM);
        ER:=Next(ER, SM);
    end;
    SM:=NIL;
    ER:=First(TSM);
    while  ER ≠ NIL do          /* third normal form */
    begin
        TFD(ER);
        ER:=Next(ER, TSM);
    end;
    RelationCompilation(SM);      /* compilation of the relationships */
    Insert(SM, SMS);      /* store the SM contents and information in SMS */
                                /* to be used in the third step */
end;   /* procedure NORMALIZATION */
```

### 3.1.3. *The CONSOLIDATION Step*

The objective of this step is to produce the final OER model by connecting all submodels. This connection is made according to the modeling rules and the classification-specific knowledge in the knowledge base. Some of the rules include : (1) merging, respectively, the identical entities and relationships whose PKs are identical, (2) creating FR's through foreign keys, and (3) creating MR's or FR’s between the submodels corresponding respectively to a superSE and its subSE according to association types (i.e., MR’s for classification types and FR’s for grouping types). All entities and relationships will be consolidated using certain heuristics to simplify their structures. The overall mapping algorithm is as follows:

```
Program SER->OER Mapping
begin
    SMS:=NIL;
    OER:=NIL;
    Input(SER);
    if exists Inheritance relationships in SER
                                /* association types are “classification” or “grouping” */
        then for each SE in SER             /* the whole hierarchy tree*/
        else  for each SE in leaf level of SER
                                        /* exists Decomposition relationships only */
            do
            begin
                SM:=NIL;
```

```
                    DECOMPOSITION(SE);
                    NORMALIZATION(SM);
                end;
        CONSOLIDATION(SMS);
        Output(OER);
    end.
```

The user (database designers) may modify the OER model derived by TSER to, e.g., obtain higher normal forms for data structures. These modifications can also be represented by virtue of the OER construct according to the definitions of OE, PR, FR, and MR. In any case, the finalized OER will, on the one hand, retain complete connection with the SER thereby allowing "zoom-in and zoom-out" between any levels of abstraction; it will also, on the other hand, lead to a complete schema design for target (or local) database systems.

#### 3.1.4. *Additional Algorithms : Models Integration, Rulebase Mapping, and Metadatabase Creation*

The above mapping algorithms can be applied to two or more functional (SER) models as well. These multiple SER models may be results of paradigm translation from object-oriented, relational, or other entity-relationship models; they may also be other TSER applications (see Figure. 3). The CONSOLIDATION step will be executed to consolidate these models two at a time, after each is completely mapped to the structural model (OER). Additional metadata such as equivalence of data items across models will be employed in the step.

On the knowledge side, the rules in Contexts and Subjects will also be related to all Entities and Relationships, which they as a whole are mapped into the rulebase model discussed in Section 3.3.

Finally, TSER extends to create the metadatabase conceptual schema (see Figure 4) for the enterprise models, and populate the metadatabase. This conceptual schema can be implemented in a number of paradigms just as any other OER model by using the algorithms discussed next.

### 3.2. *CODASYL, Relational, and Object-Oriented Models Creation*

The TSER models are translated directly into the CODASYL, relational, and object-oriented models, where only the latter needs the functional model (SER). A general framework for paradigm translation into the object-oriented models is outlined below.

**Algorithm M1** : Mapping of TSER into Object-Oriented Models.
Step 1.a : SER -> O-O (option (a))
- * Create an Object Class for every Subject, containing all attributes and knowledge.
- * Create an Object Class hierarchy corresponding to the inheritance relationships among Subjects (i.e., association types “classification” and “grouping”).
- * Go to Step 2.

Step 1.b : SER+OER -> O-O (option (b))
- * Create Object Classes from the OER model using Algorithm M2.
- * Create an Object Super-Class for each Subject, containing only an object identifier and (all) intra-subject knowledge.
- * Create a class hierarchy associating these Object Super-Classes according to the inheritance relationships among Subjects.
- * Create an inheritance (the classification type of association) between each Object Super-Class and the Object Classes created above from the OER model that corresponds to its Subject.

Step 2 : Distribution of inter-Subject rules into Object Classes.
- * Rules in every Context are incorporated into Object Classes according to the data items referenced in

their condition clauses :

* If the data items referenced in a rule belong mostly to a particular Subject, then the rule is incorporated into the Object Class corresponding to the Subject; otherwise use a tie breaker.

The structural model (OER) provides normalized data structures for implementation in any database. While directly leads to a relational model, it also fully characterizes a CODASYL model and avails a sound design for persistent objects at the foundations of objects class hierarchies.

**Algorithm M2** : Mapping of OER into CODASYL, Relational, and Object-Oriented Models.

Step 1: Convert each Relationship (along with its participating Entities) individually into the target model according to the mapping rules depicted in following graphs.

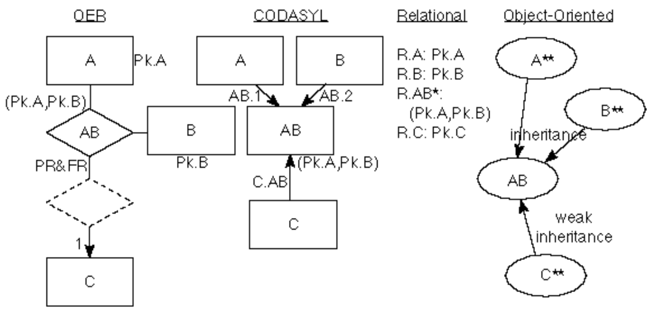
Step 2 : Declare appropriate integrity constraints according to the OER structure :

(a) For relational models: implement the key integrity rule for the OE's, and the referential integrity rule for the OR's.

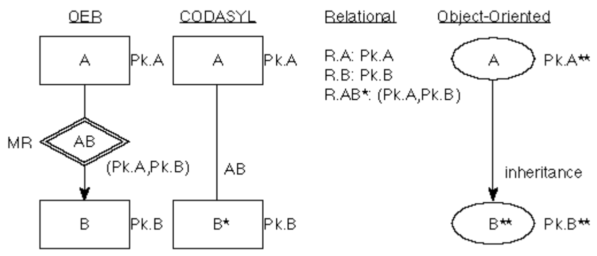
(b) For CODASYL models: implement the regular, mandatory, and fixed membership classes for FR's, PR's, and MR's, respectively.

(c) For object-oriented models: implement the association types into appropriate inheritance rules available in the models/systems.

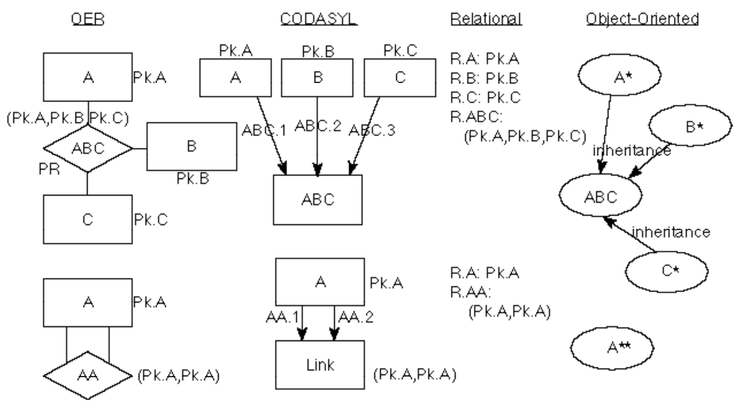
Step 3 : Implement the rules in the software environments (this step is implementation-specific).



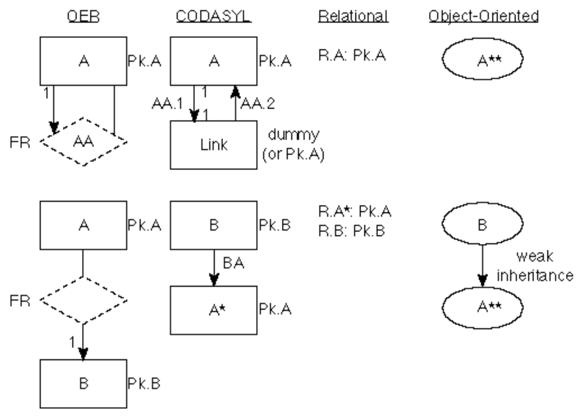
* Pk.C is included as a non-key attribute in AB.
** The attribute in Class A/B is aggregate-defined (take EXPRESS as an example, it allows aggregation type attributes) by Class B/A, meanwhile, Class C is used to define this attribute in either Class A or B.



* The membership of B in AB in the CODASYL SET is "fixed", also, R.AB is all-key.
** The membership of B is inherited, therefore is "fixed"; also, if Pk.A and Pk.B are system created primary keys, the the attribute should be eliminated, in this sense, they become all-key.



* The attribute in Class A/B/C is aggregate-defined by other two Classes; also, a class' attribute can be aggregate-defined by itself.
** A whole object hierarchy is usually entailed for recursive relationships.



* Pk.B is included as a non-key attribute in A.
** Class B is used to define an attribute in A; also, Class A's attribute can be self-defined.

## 3.3. *Rulebase Model Creation*

The rulebase model is anchored in three basic results established in the literature: (1) the metadatabase model for overall metadata integration, (2) the TSER model for information modeling primitives (metadata classes), and (3) the expression grammar, along with predicate logic, for production rule structures. The basic approach used in modeling knowledge in the metadatabase combines ideas from production rules, frames, and object-oriented representations, through the TSER modeling methodology.

The basis of this derivation is the specific grammar of rules that we constructed from general results in the field. It is shown as $G_1$ and $G_2$ below.

$G_1$: (1) <Rule> ::= IF <Condition> THEN [<Action>]$^+$
(2) <Condition> ::= <Expression>
(3) <Action> ::= <Declarative-Statement> | <Assignment-Statement>| <Procedure-Call>
(4) <Assignment-Statement> ::= <Action-Ident> := <Evaluated-Fact>
(5) <Procedure-Call> ::= Procedure-name([<Parameter-List>]$^*$)

$G_2$: (1) <Expression> ::= <Fact> <Operator> <Fact> | <Fact>
(2) <Fact> ::= <Evaluated-Fact> | <Declarative-Statement>
(3) <Evaluated-Fact> ::= <Simple-Fact> | <Composed-Fact>
(4) <Simple-Fact> ::= Constant | Item | <User-Ident> | <Action-Ident>
(5) <User-Ident> ::= Identifier
(6) <Action-Ident> ::= Identifier
(7) <Composed-Fact> ::= <Function-call> | <Expression>
(8) <Function-call> ::= Function-name([<Parameter-List>]$^*$)
(9) <Parameter-List> ::= <Parameter>[, <Parameter-List>]$^*$
(10) <Parameter> ::= <Fact>
(11) <Declarative-Statement> ::= <Simple-Fact> Verb <Simple-Fact>

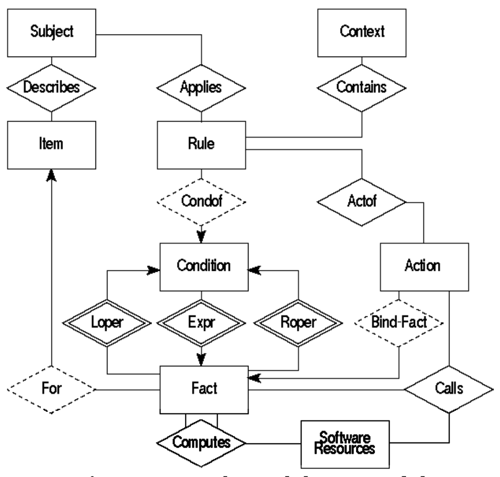


**Figure 5.** *The Rulebase Model*

TSER constructs are employed to represent the different components of the rulebase according to G1 and G2, and are consolidated into an overall normalized model shown in Figure 5, which is, in turn, incorporated into meta-model in Figure 4. The result of this consolidation also constitutes a generic structure for the rulebase model. This model can be implemented using commonly available database technology (such as relational databases) and routines/files manager plus an inference engine [BOU 91]. The active aspects of this model, including rules management and operation in distributed environments, are fully supported by the new ROPE (rule-oriented programming environment) technology as discussed in [BAB 93]. Their details, however, are not included in the space of this paper.

## 4. Paradigm Translation : The PDES/EXPRESS Case

EXPRESS is an information modeling language developed for product design and engineering databases under the auspices of ISO/PDES endeavors. It combines the traditional Entity-Relationship paradigm into the latest object-oriented paradigm with rules [SAN 92, SMI 89]. The design also reflects a mix of data modeling construct with certain software engineering concepts for programming. As a result, EXPRESS is more powerful and complicated than one would expect for an object-oriented data definitional language. A reverse engineering success on EXPRESS paves the way for similar efforts on other object-oriented systems and entity-relationship models in the field.

EXPRESS is based on an extensible object-oriented semantic association knowledge model. It supports the basic association types of generalization and aggregation among the primitive objects, called entities. Generalization association is specified as "SUPERTYPE <classes/entities>" or "SUBTYPE <classes/entities>", and aggregation association is in the form of "instance_variable <aggregation>, key <identifier>." The class/entity properties can be inherited through the generalization association. Behavioral properties of a class/entity are specified as methods and rules. Also, in EXPRESS, there are two major categories of item definition. One is the item defined by **non-aggregation** type, i.e., items declared as real, integer, Boolean, Entity...etc. The other is the items defined by **aggregation** type, i.e., items declared as array, list, bag, set...etc. While the first category is straight forward, the second can be used to imply aggregate-defined entity/object hierarchy. This category actually represents multiple/plural relationships among entities from a usual data modeling perspective. The inheritance mechanism is the major characteristic of object-oriented concept. In EXPRESS, there are two ways to define the inheritance relationship between entities. The first is by using the keyword SELECT, which allows a child-entity's attributes inherited from one of its parent-entities. The second is by using the keyword SUBTYPE or SUPERTYPE, which allows a child-entity's attributes inherited from several parent-entities.

A general algorithm mapping an EXPRESS model into the TSER model is given below.

**Algorithm RM1** : EXPRESS -> TSER

Step 1 : Convert the object definition into Subject.

- * Create a Subject for each Entity.
- * Create a Subject for each Type/Selection Type.

Step 2 : Model the specific item definition.

- * Create Placeholder Identifier for each Subject if there is no UNIQUE (key declaration) item(s) in Entity.
- * Create same items in Subject for those items declared through non-aggregation type in Entity.
- * Declare FR's for those items declared through non-base type in Entity; also declare those items equivalent to non-base type's key
- * Declare PR's for those items declared through aggregation type in Entity. Declare FD's as "(key, aggregation type's key) <-> (key, aggregation type's key)" for PR-related items; and as "key -> other non-key items" in Subject.

Step 3 : Model the inheritance constructs.

- * Duplicate Super-Subject's key in Sub-Subject for inheritance relationships (SUPERTYPE/SUBTYPE, or SELECTION).
- * Declare FD's as "Sub-Subject's key -> Super-Subject's key".

Step 4 : Model the methods/rules and other special definitions.

- * For methods that return a value : define a data item representing each method and model the method itself as an expression to be fired by a rule, according to the grammars of the rulebase model in Section 3.3;
- * For other methods : model them as rules with all variables classified into data items that correspond to Subjects (names and attributes) and those that are other facts (also according to the rulebase model).
- * Create from methods and rules Inter-Subject rules for each Subject, according to the SUBJECT

definition.

* Create from methods and rules Intra-Subject rules in Contexts, according to the CONTEXT definition.
* Identify UNIQUE keyword and update declared items, FD's, equivalence table in Subject if necessary.
* Identify INVERSE keyword and update PR/FR declaration if necessary.

To illustrate this procedure, we first present an EXPRESS schema taken from a sheet metal design database, then show the result of its reverse engineering into TSER. Several comments follow. As shown in the example, each object/entity (e.g. Feature and Point_2D) is mapped to a Subject at the functional level. Most instance_variables (e.g., feature_ID, x, y, etc.) are attributes of a Subject. When converting the semantic meaning for the inheritance constructs, the generalization association is represented by a mandatory relationship (MR) and parent-entity's key (e.g., Feature's feature_ID) is added as a foreign key to child-entity (e.g., Close_Loop). Concerning methods (not shown in the above EXPRESS example) that return a value can be treated as derived data items and the function itself will be referred to by firing a rule. All other methods and rules can be mapped to operating rules of a Subject. Some special definitions should also be considered while converting EXPRESS into other models. One such definition is the keyword UNIQUE, which allows an object to have a key in entity, thus, can be used to define functional dependencies and the primary key for a Subject. Another is the keyword INVERSE, which allows some relationship constraints between entities, thus, implies necessary relationship modifications between Subjects.

## The Example of Sheet Metal EXPRESS Model

```
SCHEMA SheetmetalObjects;

ENTITY Feature;
        feature_ID: INTEGER;
        UNIQUE single: feature_ID;
END_ENTITY;

ENTITY Point_2D;
        x: Real;
        y: Real;
END_ENTITY;

ENTITY Line;
        start_Point: Point_2D;
        end_Point : Point_2D;
END_ENTITY;

ENTITY Arc;
        center_Point: Point_2D;
        start_Point : Point_2D;
        end_Point : Point_2D;
END_ENTITY;

TYPE
        geo_Type = SELECT(Line,Arc);
END_TYPE;

ENTITY Closed_Loop
        SUBTYPE OF (Feature);
        geo_List: LIST\[0:100] OF geo_Type;
END_ENTITY;

...

ENTITY Sheetmetal_Feature
        SUBTYPE OF Closed_Loop);
        centerpoint : Point_2D;
        theta_Rot : REAL;
        datum_Feature : Feature;
        location_Tol : Loc_Tol_2D;
        gT_Position_Flag: BOOLEAN;
        gT_Position_Tol : REAL;
END_ENTITY;

ENTITY Loc_Tol_2D;
        x_Tol_Plus : REAL;
        x_Tol_Minus: REAL;
        y_Tol_Plus : REAL;
        y_Tol_Minus: REAL;
END_ENTITY;

...

ENTITY Sheetmetal_Part;
        plate_Charact : LIST 1:?] OF Sheetmetal_Feature;
END_ENTITY;

ENTITY Product_Version
        product : Product;
END_ENTITY;

ENTITY Product;
        product_ID : STRING(5);
        components: LIST [1:?] OF
        Sheetmetal_Part;
        UNIQUE single : product_ID;
        INVERSE versions : SET[1:?] OF
        Product_Version FOR product;
END_ENTITY;

END_SCHEMA;
```

## The TSER results converted from the Example of Sheet Metal

## Schema

```
SUBJECT Feature
ITEMS: feature_ID : INTEGER;
FD's : feature_ID <-> feature_ID;
END_SUBJECT;

SUBJECT Point_2D
ITEMS: $Point_2D : String;
        x : Real;
        y : Real;
FD's : $Point_2D -> (x, y);
END_SUBJECT;

SUBJECT Line;
ITEMS: $Line : String;
        $geo_Type : String;
        start_Point : String;
        end_Point : String;
FD's: $Line -> (start_Point, end_Point);
        $Line -> $geo_Type; // MR efinition
Declare : FR
        Two FR between (Line, Point_2D) due to items
        "start_Point" and "end_Point"
        Equivalence :
            (start_Point, $Point_2D)
            (end_Point, $Point_2D)
END_SUBJECT;

        ...

SUBJECT geo_Type
ITEMS: $geo_Type : String;
FD's: $geo_Type <-> $geo_Type;
END_SUBJECT

SUBJECT Closed_Loop
ITEMS: $Closed_Loop : String;
        feature_ID : INTEGER;
        $geo_Type : String;
FD's: ($Closed_Loop, $geo_Type) <->
        ($Closed_Loop, $geo_Type)
        $Closed_Loop -> feature_ID;  (MR)
Declare : PR
        name of all key relation are "geo_List"
END_SUBJECT

...

SUBJECT Sheetmetal_Part
ITEMS: $Sheetmetal_Part : String;
        $Sheetmetal_Feature : String;
FD's:
($Sheetmetal_Part, $Sheetmetal_Feature)
<-> ($Sheetmetal_Part, $Sheetmetal_Feature)
Decalare : PR
        name of all key relation are "plate_Charact"
END_SUBJECT

SUBJECT Product_Version
ITEMS: $Product_Version: String;
        product: String;
FD's: $Product_Version -> product;
Declare: FR
        One FR between (Product_Version, Product) due
        to item "product"
Equivalence :
        (product, product_ID)
END_SUBJECT

SUBJECT Product
ITEMS: product_ID: String;
        $Sheetmetal_Part : String;
        versions : String;
FD's: product_ID -> versions;
        (product_ID, $Sheetmetal_Part) <-> (product_ID,
        $Sheetmetal_Part)
Declare : PR
        name of all key relation is "components"
Equivalence :
        (versions, $Product_Version)
** product_ID becomes the key of this subject due to
        "UNIQUE" definition
** one PR has been replaced to be FR due to
        "INVERSE" definition
END_SUBJECT
```

A structural model (OER) of the object example is shown in Figure 6. Note that some of the relationships do not correspond directly to the original objects; but rather, they represent certain semantic information derived from objects that need to be represented in the metadatabase. For instance, start_Point in Line and Arc is associated with the equivalence definition start_Point = Point_2D. Similarly, plate_Charact in Sheetmetal_Part serves as a representation of the association between sheetmetal parts and features.

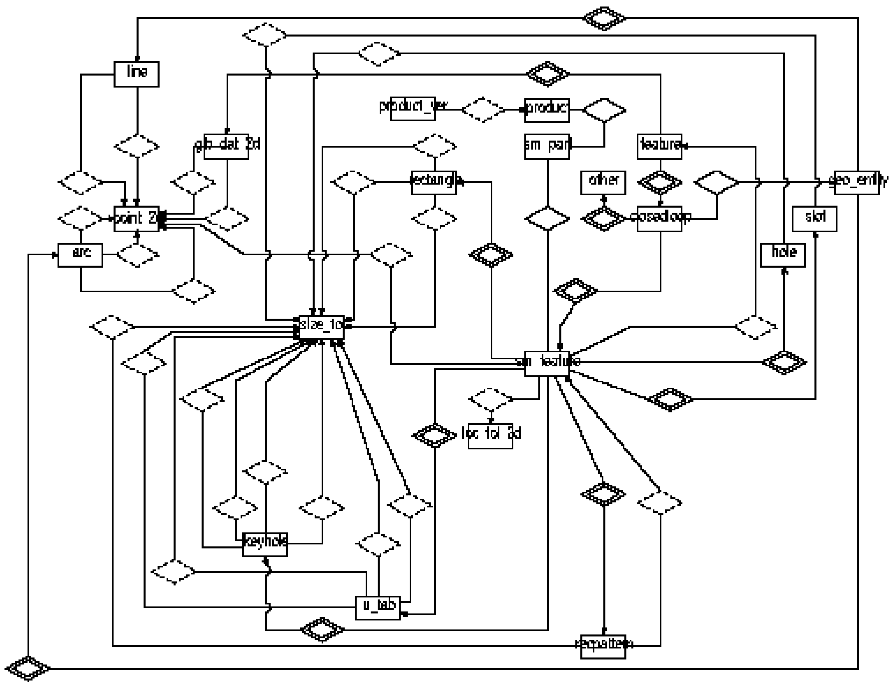


**Figure 6.** *OER for Sheetmetal Model*

## 5. Manufacturing Information Integration Using Meta-Models

The TSER approach to manufacturing meta-models calls for an information modeling environment and a metadatabase system, as discussed in Section 1. Both have been prototyped in an integrated manufacturing laboratory at Rensselaer. Previous results of the prototypes facilitating integration are reported in [BOU 91, CHE 91, HSU 92], continual research in the light of Figures 1 and 2 is ongoing. At present. all algorithms and models discussed in this paper have been implemented, and their completeness and correctness in actuality are under empirical investigations with satisfactory results.

The modeling prototype is coded mainly in C and implemented on three platforms : IBM RS6000, DEC station, and PC. It performs the functions necessary to support the first two levels (for modeling and models integration) of meta-model, with the third level (for information management) achieved through the metadatabase prototype. Both prototypes are connected to gain maximal leverage on each others, but can function perfectly in their own.

The metadatabase system has been implemented on a MicroVAX platform using Digital's Rdb as the database engine for the metadatabase. A new version using the IBM AIX RS/6000 workstation and Oracle DBMS has recently been completed to provide a multi-platform and distributed metadatabase environment. To facilitate user and program interactions with the metadatabase, a shell has been developed in the C language. Currently, enterprise users interact via a menu (in the VAX version) or an X-Windows/Motif® (in the AIX version) interface, while other systems interact through a metadatabase query language application program interface (API). These other systems are manufacturing application systems including Product Design using PDES/EXPRESS model implemented on ROSE/RS6000, Shop Floor Control on ORACLE/PC, Process Planning on dBase/PC, and Order Processing on Rdb/MicroVAX.

In sum, a compact meta-model solution is provided in this paper. The basis of this solution, the TSER approach, is designed to be simplistic while accomplishing the necessary requirements of a meta-model method. At present, the TSER approach supports rule-based systems, object-oriented models, Entity-Relationship models, relational systems, CODASYL systems, and flat file systems in both concepts and methods; its software implementation includes automatic paradigm translations for EXPRESS and SQL and metadatabase creation capabilities for MicroVAX using Rdb and RS6000 using ORACLE. Its overall ontology is compatible with the spirit of the IRDS standards [HSU 91]. Ongoing research includes further development of paradigm translations for particular models, database systems, and CASE tools. Development of a reference model (in the sense of [ESP 89, RAT 90]) with economical evaluation capability to guide integration modeling is an extension to the meta-model system; which is currently underway.

As a final remark, we might mention three contributions of this paper: the concept of meta-models, a complete model of its implementation using a metadatabase, and a tested modeling environment for supporting the needs of meta-modeling in a manufacturing enterprise. Furthermore, the compactness of TSER, as evidenced in its implementation, is a significant advantage compared to other methods for models integration.

## 6. Acknowledgment

This research is supported in part by Alcoa, DEC, GE, GM and IBM through the CIM and AIME research programs at Rensselaer, and NSF grant DDM-9015277.

## 7. References

[ANS 85] American National Standards Institute. (Draft proposed) American National Standard Information Resource Dictionary System: Part 1-Core Standard. ANSI X3H4, American National Standards Institute, New York. 1985.

[BAB 93] G. Babin, *Adaptiveness in Information Systems Integration*, unpublished Ph.D. Thesis - Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, August 1993.

[BOU 91] M. Bouziane, “Metadata Modeling and Management,” unpublished Ph.D. Thesis - Computer Science, Rensselaer Polytechnic Institute, June 1991.

[CHE 91] W. Cheung, “The Model-Assisted Global Query System,” unpublished Ph.D. Thesis - Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, November 1991.

[DOL 87] D. Dolk and R. Kirsch, II. "A Relational Information Resource Dictionary System," *Communications of the ACM*, Vol. 30, No. 1, January 1987, pp. 48-61.

[ESP 89] ESPRIT Consortium AMICE (eds.), *Open System Architecture for CIM,* Springer-Verlag, 1989.

[FUL 92] J. Fulton, "Enterprise Integration Using Semantic Unification," *Proceedings of 1st International Conference on Enterprise Integration*, MIT Press, October 1992.

[HSU 85] C. Hsu, "Structured Database Systems Analysis and Design Through Entity-Relationship Approach," *Proceedings 4th International Conference on Entity-Relationship Approach*, IEEE Computer Society, 1985, pp. 56-63.

[HSU 87] C. Hsu 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.

[HSU 90] C. Hsu and L. Rattner, "Information Modeling for Computerized Manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 4, 1990, pp. 758-776.

[HSU 91] C. Hsu, M. Bouziane, L. Rattner, L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," *IEEE Transactions on Software Engineering*, June, 1991, pp. 604-625.

[HSU 92] C. Hsu, 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.

[HSU 93a] C. Hsu, L. Gerhardt, D. Spooner, A. Rubenstein, " Adaptive Integrated Manufacturing Enterprises: New Information Technology for the Next Decade," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 33, No. 3, 1993.

[HSU 93b] C. Hsu and G. Babin, "A Rule-Oriented Concurrent Architecture to Effect Adaptiveness for Integrated Manufacturing Enterprises," in *International Conference on Industrial Engineering and Production Management*, Mons, Belgium, pp. 868-877, June 1993.

[HSU] C. Hsu, G. Babin, M. Bouziane, W. Cheung, L. Rattner, A. Rubenstein, and L. Yee, "The Metadatabase Approach to Integrating and Managing Manufacturing Information Systems," *Journal of Intelligent Manufacturing*, forthcoming.

[PAT 92] R. Patil, R.Fikes, P. Patel-Schneider, D. Mckay, T. Finin, T. Gruber, and R. Neches, "The DARPA Knowledge Sharing Effort : A Progress Report," *Proceedings the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Massachusetts, October 1992.

[RAT 90] L. Rattner, "Information Requirements for Integrated Manufacturing Planning and Control: A Theoretical Model," unpublished PhD Thesis - Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, November 1990.

[SAN 92] D. Sanderson and D. Spooner, *Mapping Between EXPRESS and the Extended Entity Relationship Model*, Proceedins of the EXPRESS User Group Meeting, Houston, TX, 1992.

[SHE 90] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, And Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, 1990, pp. 103-136.

[SMI 89] B. Smith, Product Data Exchange Specification : First Working Draft, NISTIR BB-4004, NIST Order number PB 89-144794, 1989.

[SOW 91] J. Sowa, "Towards the Expressive Power of Natural Languages," in J. Sowa (ed.) *Principle of Semantic Networks*, Morgan Kaufmann, 1991, pp. 157-189.