EEP-Vol. 19-1, Advances in Electronic Packaging -- 1997
Volume 1, ASME 1997

MODELS OF ELECTRONIC PACKAGE ENGINEERING

Fred L. Cox III
Georgia Tech Research Institute
Atlanta, Georgia

Gintautas B. Jazbutis
Structural Dynamics Research Corporation
Seattle, Washington

ABSTRACT

Some CAD and CAE tools exist to support electronic package engineering. However, these tools address only part of the needs of the discipline. Many of these tools are merely adaptations of tools developed for other areas, such as printed circuit board design, and do not address the unique needs of single- and multi-chip modules nor allow for the increased complexities of the advanced packaging technology now being developed. Further cost reduction could be realized by providing consistent, unified information and data management throughout the design, fabrication, assembly, and testing of packages.

Formal models of the electronic package engineering process are needed to guide the development of software and databases to achieve these design automation goals. This paper reports on the development of such models, written in IDEF0 and EXPRESS. These models should be useful in determining the needs for CAD and CAE software in the package engineering process and in establishing requirements for software integration and interoperability.

OVERVIEW

The models presented in this paper were developed under support provided by the Georgia Tech Packaging Research Center (PRC). A key objective of the Packaging Research Center is to create the technology required for the development of low cost, high performance electronic packages. The use of computer aided design and design automation tools has proven critical to reducing cost in related areas of engineering, such as the design of integrated circuits. With the emphasis on high performance packaging comes an increased complexity of the packages themselves, which in turn increases the need for tools that automate or aid package engineering activities. The new electronic packaging technology also has a need for much greater integration of the tools supporting electrical, mechanical, and thermal design.

In recognition of these needs and opportunities, the Packaging Research Center formed the Design Automation and Integration Thrust Team to address these issues. It became clear that a formal description of the electronic package engineering process, from specification and design through testing, was needed to support adequate definition of the requirements for design automation. Consequently, the development of process and information models was begun, with the initial effort being focused on the areas of design and testing. The results of this effort are reported in this paper.

The next section will describe some of the problems encountered in design automation for electronic package engineering and their consequences. The following section will discuss the needs for formal models, how the models should be used, and the manner in which the models were developed. The next two sections provide information on how to interpret the models, including introductions to the formal languages used to describe the models. Recommendations are then given for further research and use of the models. Some representative models are also provided to give the reader an idea of the development and use of such models.

PROBLEMS IN DESIGN AUTOMATION FOR PACKAGING

There are a number of problems related to software for supporting electronic package engineering as well as in the engineering process itself. These problems appear to be significant cost drivers that need to be solved. Some of these problems are discussed in this section.

As noted earlier, some software tools and tool packages exist for computer aided design of electronic packaging. However, these tools

address only a subset of the needs of the area. Further, the typical package engineering organization finds itself dependent on an assortment of tools from various vendors, since some tools more effectively meet the organization's specific needs than others. It is not unusual for the mix of tools to include software developed in-house to provide functionality not supplied by commercially available tools. While any given vendor's toolset may be reasonably well integrated, it is rare to find adequate integration for tools across multiple vendors. There is also a low level of tool interoperability (the ability of tools to share data), which results largely from the lack of common data formats and the dependence of tools on vendors' own proprietary data formats.

These problems in integration, interoperability, and unavailable functionality place a significant burden on electronic package engineers, forcing them to patch together multiple software tools and manual processes. This complexity adds to the steep learning curve typically associated with individual tools and results in greater expense for training and retraining of personnel. Ideally, automation and aiding tools should reduce the level of expertise required of engineers, making it possible to staff the operation with less highly qualified individuals. However, the current situation increases the level of qualification required by adding the need for expertise in use of the tools as well as in navigation of the patchwork of automated and manual operations. The interoperability problem can also force the reentry of data or the development of special data format translators, which in turn often lead to data inconsistencies as well as redundant storage of data.

The problems generally are accompanied by poor management of information and data, even in single-vendor tool frameworks. Typically, there is little, if any, version control available. There is rarely any support for tracking the relationships among data objects, such as file derivation, design splitting, or the association of simulation results with the particular version of a design giving rise to them. This valuable function is left to be manually implemented by the engineers, who often ignore it.

Loss of information also occurs frequently. For example, much of the information needed by test engineers will normally have been generated during the design process but discarded. As a result, the time and cost of test development is increased by the need for test engineers to regenerate the lost information. Regeneration of the information also lends itself to the creation of inconsistencies and consequent testing problems. This situation also highlights a common problem in package engineering, the lack of concurrency in design and test development. Properly coordinating these two activities and making them concurrent can reduce problems with testability, reduce the delay between the ability to fabricate packages and the ability to test them, and reduce the number of design iterations, thereby shortening the time to market.

TECHNICAL APPROACH

In defining requirements for software to support electronic package engineering, it is important to determine what information is created in the engineering process and the general information dependencies, binding constraints, and flow. It is also important to determine the functional coverage provided by existing software, to evaluate the quality of that coverage, and to identify any inadequacies. Given a cost profile of the overall engineering process, this knowledge can provide a basis for projecting the potential economic benefit and return on investment of providing various forms of automation, aiding, and integration. Such an analysis, in turn, would support the prioritization of software development efforts.

A formal means of representation is needed for describing the overall package engineering process and its information content, both as it exists and as it might be preferred. IDEF0 and EXPRESS provide such representational formalisms and have been used productively to improve automation and integration in such domains as mechanical engineering. IDEF0 supports the formal description of the functional activities of a process, along with their interrelationships, resource requirements, inputs, outputs, and control factors. EXPRESS enables the formal representation of information elements, their attributes, relationships, and associated constraints. EXPRESS may be used to provide detailed representations of the components of an IDEF0 model. Together, these formalisms provide a powerful mechanism for portraying the engineering process in a manner that supports communication, analysis, and common rules of interpretation. In a sense, these tools are to process engineering what schematics and component data sheets are to electronic circuit design.

Models in IDEF0 and EXPRESS can provide a clear, objective basis for mapping the functionality of existing software tools and frameworks onto the engineering process to identify gaps in coverage and specific problems with integration and interoperability. Models of this kind can also help identify opportunities for improving the engineering process, such as increasing the concurrency of constituent activities, as well as aid in ascertaining inherent constraints on such changes. Other potential benefits of these models include:

- help in determining optimal information creation, use, and storage, so as to avoid information loss, redundancy, or inconsistency;
- aid in evaluating the effects of alternative design methodologies;
- ready identification of integration and interoperability requirements and the need for standard data formats and protocols;
- automatic generation of shared databases;
- support for training persons unfamiliar with the package engineering process or aspects of it; and,
- facilitation of communication for persons and organizations attempting to analyze and improve package engineering.

In developing these models, a decision had to be made whether to model a specific process, such as that of a particular organization, or to develop models for a generic process. For the current project, a generic approach was deemed more appropriate for PRC purposes and more consistent with the level of funding and availability of experts from organizations involved in electronic package engineering. (One organization pointed out that it considered its engineering process to be a key factor in the organization's ability to compete and consequently was unwilling to share this proprietary information.) The generic models thus developed should provide a reasonable basis for evaluating the functional coverage and integration of existing software. They should also provide examples and starting points for any organization wishing to develop models of its own engineering process.

Another decision regarded the use of one or many expert sources. With either approach, it is helpful to start with a single expert, since

doing so reduces problems with convergence and leads to a concrete draft early on to which others can respond. In an engineering organization, the broad review of such a draft or strawman can produce very valuable insights as well as highlight differences of view and alternative preferences. Although surfacing these differences can prove interesting, it is a critical step toward achieving a unified view of the process, shared throughout the organization. This step is also fundamental to improving the process, since little progress can be made when communication and cooperation are undercut by unsuspected, divergent views of the process as it exists. Also, potential expert sources sometimes perceive a threat to the security of their jobs when asked to participate in developing models to be used in supporting automation. However, the attendant reluctance to participate may often be overcome by the desirability of illuminating problems in the process as it exists and offering ways to improve it.

In this project Dr. Madhavan Swaminathan of the Packaging Research Center was the primary subject matter expert, who gave liberally of his time and knowledge. Information regarding design parameters was provided by Dr. Timothy J. Drabik and Albert Titus. Information on testing was provided by Bruce Kim and Sasidhar Koppolu. We also drew information from written sources, such as the Microelectronics Packaging Handbook [Tummala and Rymaszewski]. Any errors, inadequacies, or misinterpretations in the models should be attributed to the authors and not to these sources. However, it is to be expected that anyone familiar with electronic package engineering will entertain divergent opinions regarding at least some aspects of the models.

IDEF0 MODEL OVERVIEW

The IDEF modeling methods are a product of the United States Air Force's Integrated Computer Aided Manufacturing (ICAM) program of the late 1970's and early 1980's. The primary goal of the program was to increase productivity; the ICAM approach was to establish structured methods and tools for applying computer technology to manufacturing. The ICAM Definition (IDEF) family of methods was defined to better understand, communicate, and analyze manufacturing [Bravoco and Yadav].

The ICAM program established a System Development Methodology, the components of which are:

- a management approach to integrated systems development which entails four major steps: understanding the problem, formulating the solution, building the solution, and implementing the solution;
- a standard mechanism for the development and documentation of the technical analysis, comprised of the IDEF0, IDEF1, and IDEF2 methodologies; and,
- a common approach to developing and validating system models.

The IDEF methods are used to provide a structure around which the tools and methodologies for integrated systems development can be constructed.

IDEF0 is a modeling method by which the activities and functional aspects of a process may be represented. The models graphically describe the flow of items among activities, the resources used in the activities, and constraints on the activities. The method is top-down and hierarchical, and models can be developed to a high degree of detail. The first level (one-page drawing) of each model generally contains one activity. This activity may be decomposed into greater detail by creating a new level containing multiple activities. Each activity in turn may be decomposed further, until the desired level of detail is achieved. Figure 1 shows the hierarchical structure used in IDEF0.

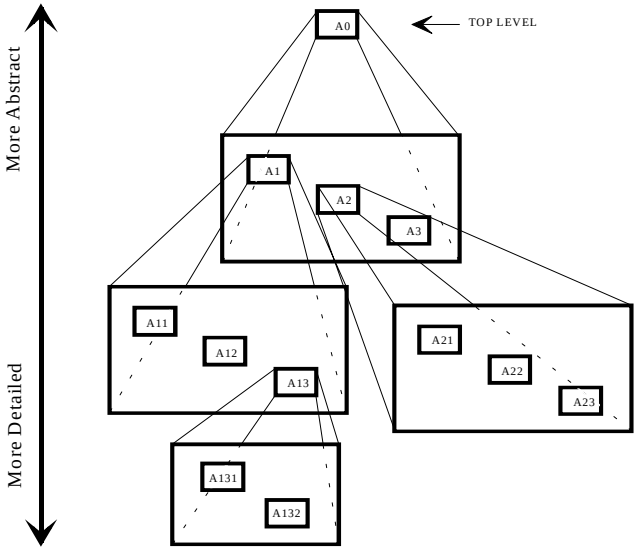

FIGURE 1. IDEF0 HIERARCHICAL STRUCTURE.

An IDEF0 diagram consists of the following components (Figure 2):

- *Activities*. Activities (system functions) are represented by boxes and have a verb descriptive title.
- *Inputs*. Inputs are represented by arrows entering the left side of an activity. Inputs are items upon which the activity works.
- *Outputs*. Outputs are represented by arrows exiting the right side of an activity. Outputs are items that are a result of the activity.
- *Controls*. Controls are represented by arrows entering the top of an activity box. Controls are things that influence the activity, such as specifications or constraints.
- *Mechanisms.* Mechanisms are represented by arrows entering the bottom of an activity box. Mechanisms are resources needed to perform the activity, such as personnel or machinery.
- *Node Numbers*. Node numbers designate the location of activities in the hierarchy. A0 designates the top-level activity; A0 decomposes to A1, A2, A3, and so forth; A1 in turn decomposes to A11, A12, A13, and such. The node number indicates the location of each activity in the hierarchy.

Text and *glossary*. Each page of a diagram may have some descriptive text, and a glossary may exist to define items and terms used in the model.

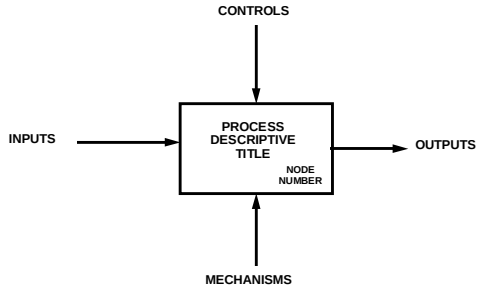

FIGURE 2. BASIC IDEF0 SYNTAX.

Once an IDEF0 model has been implemented, the model developers will have studied, understood, and characterized the activities and functions of the overall process; they will also have identified key events in the process. While most people in an organization may have detailed knowledge of how their particular area works, few know how all areas work. IDEF0 provides a method to collect, characterize, and communicate this information. A well developed IDEF0 model may also highlight subprocesses that need improvement. In addition, a "To-Be" IDEF0 model may developed to describe how the process *should* be; such a model would provide a target for process improvement.

IDEF0 Model of Electronic Package Engineering

Portions of the IDEF0 model developed for this project are found in Figures 3 through 6. In addition to the models themselves, a glossary of all the arrows was developed. A brief description of each figure follows.

Figure 3: This figure provides the top-level diagram and represents the overall activity of electronic package engineering: produce electronic packages.

Figure 4: This figure shows the first level decomposition of the A0 activity into the five primary activities of developing packages:

- development of product description and specification;
- package design;
- package fabrication and assembly;
- interconnect and functional test;
- package rework.

Figure 5: This diagram details the A2 activity. Initially, a set of rules are developed (testing, thermal, mechanical, and electrical) which constrain the physical design of the package. The package is then designed, verified for adherence to the rules, and post-processed.

Figure 6: This figure breaks out the interconnect and functional test of the assembled package. Each in turn decomposes into more detail. A major feature of the functional test is the Built-In Self-Test (BIST) capability of the package product.

INFORMATION MODEL OVERVIEW

EXPRESS is a language and a method for information modeling developed under the STEP (**ST**andard for the **E**xchange of **P**roduct data) program, an international effort to standardize product data formats to improve interoperability. EXPRESS is a textual language; however, it has a corresponding graphical representation, called EXPRESS-G, which supports a subset of the semantics of EXPRESS and can be generated from an EXPRESS model. EXPRESS-G can also be used independently of EXPRESS [ISO TC184/SC4/WG5]. EXPRESS-G represents entities and their relationships to each other and incorporates the inheritance feature of object-oriented design. EXPRESS-G can also describe attributes of entities, but, unlike EXPRESS, it does not represent operations on the attributes of entities.

An EXPRESS-G diagram consists of the following components (Figure 7):

- *Entities*. An entity represents anything that can be considered a physical or logical object. Entities are represented graphically by rectangles.
- *Relationships*. Entities may be related to one another in various ways. EXPRESS-G focuses on one-way relationships: a circle designates the "to" end of a relationship. Relationships may be of three kinds:
  - *normal* (depicted as a normal line),
  - *tree* (indicates inheritance of a supertype and its subtype; depicted as a thick line), and
  - *optional*, which shows a weak or optional or schema-to-schema relationship(depicted as a dashed line).
- *Simple Type Symbols*. Represented by rectangular boxes with a double vertical line on the right side, these are the predefined types available in EXPRESS.
- *Type Symbols*. Represented by dashed-boxes, there are three forms of types: SELECT, ENUMERATION, and user-defined types.
- *Page References*. An EXPRESS-G diagram may span more than one page. Page references allow the modeler to refer to entities on the same or other pages.
- *Inter-Schema References*. Another schema (model) may be referenced using components of this type.

The reader is referred to the *EXPRESS Language Reference Manual* for a full description of the EXPRESS lexical modeling syntax [ISO TC184/SC4/WG5, Part 11] .

EXPRESS supports the object-oriented approach in information systems design and development. Entities may have a number of attributes, and they may define operations valid on these attributes. This approach allows incremental development and implementation of information systems and supports code reuse. Other approaches that have been proposed for object-oriented design include Yourdon's, Rumbaugh's, and IDEF4.

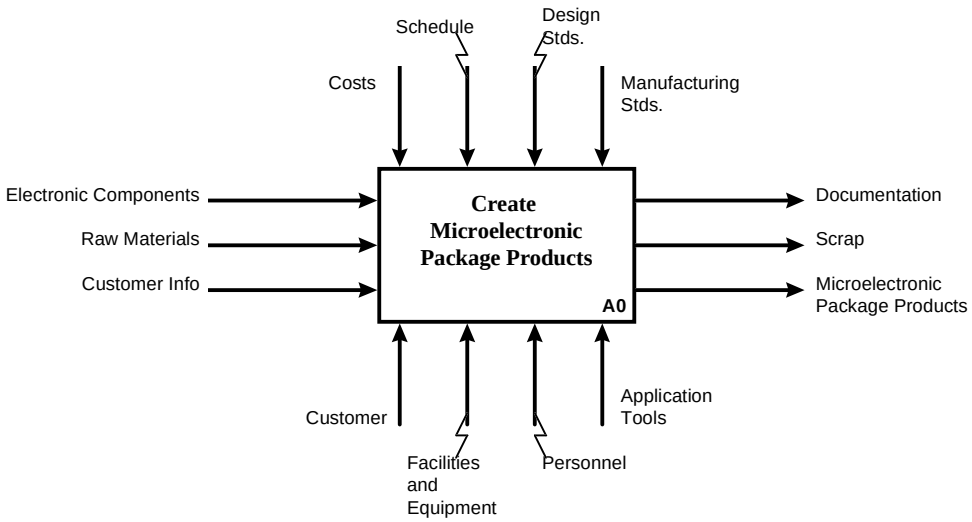

FIGURE 3. TOP LEVEL DIAGRAM.

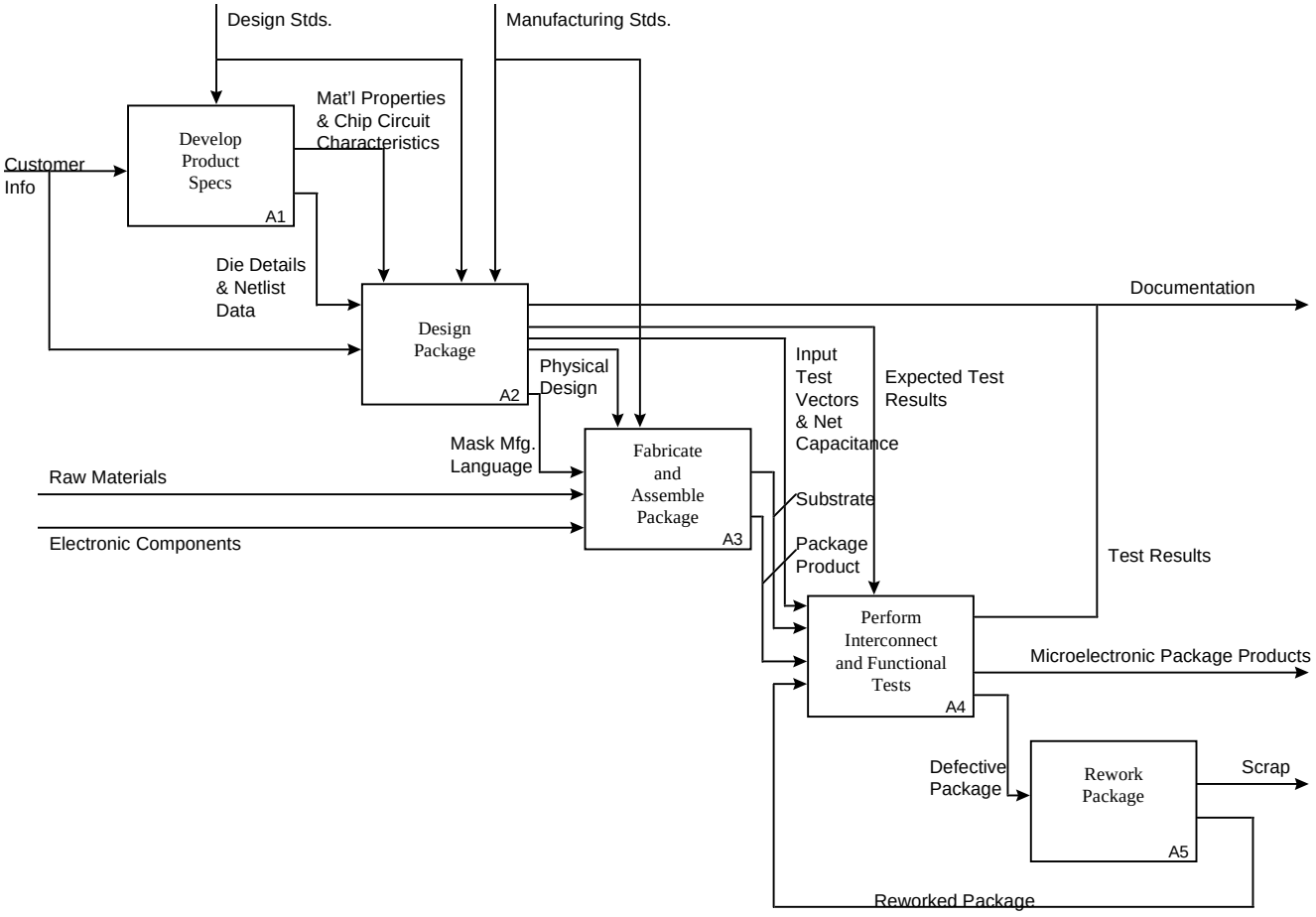

FIGURE 4. DECOMPOSITION OF A0 ACTIVITY BOX.

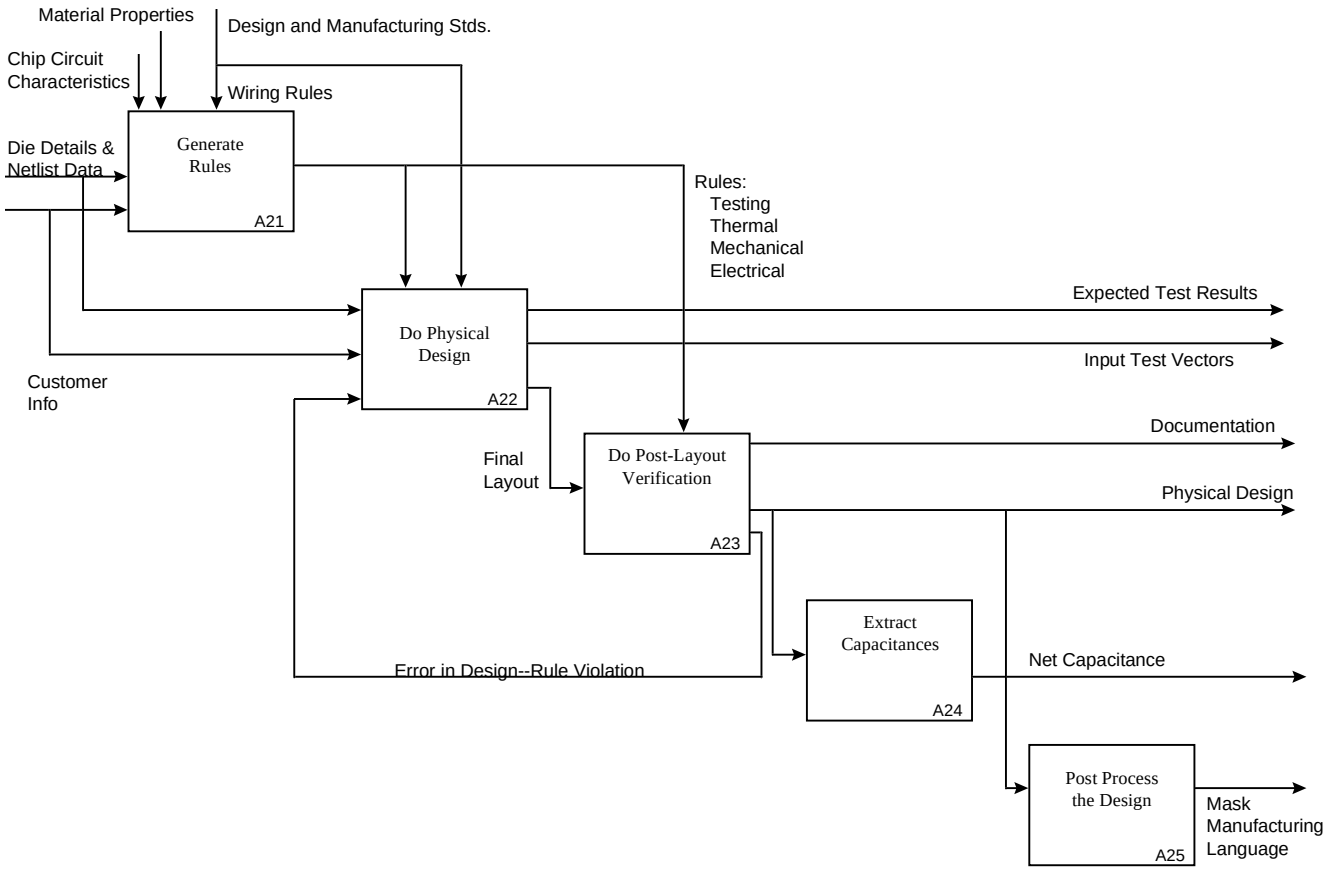

FIGURE 5. DECOMPOSITION OF A2 ACTIVITY BOX.

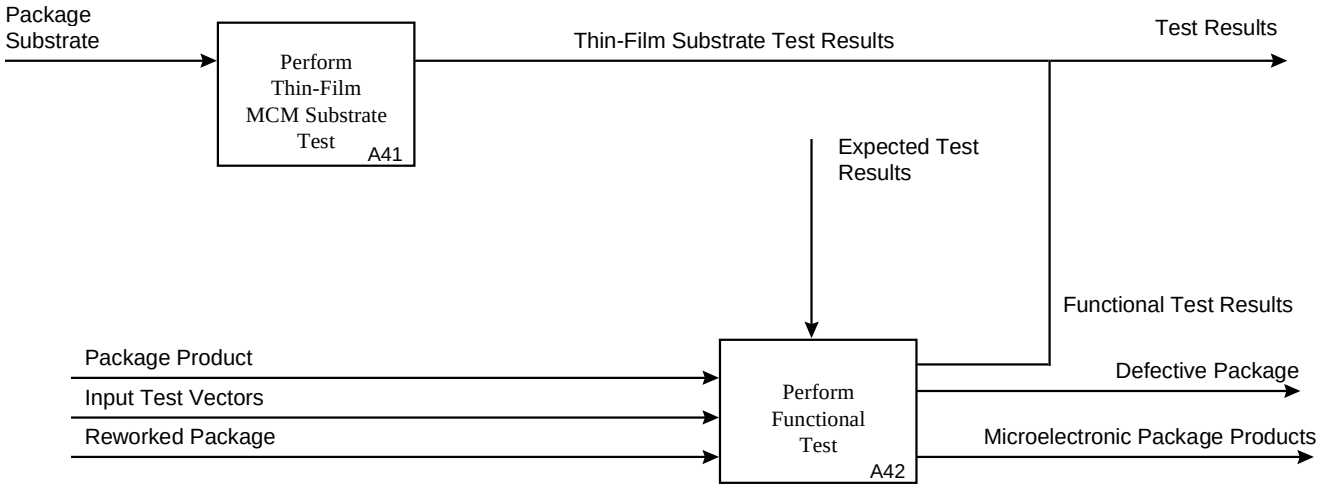

FIGURE 6. DECOMPOSITION OF A4 ACTIVITY BOX.

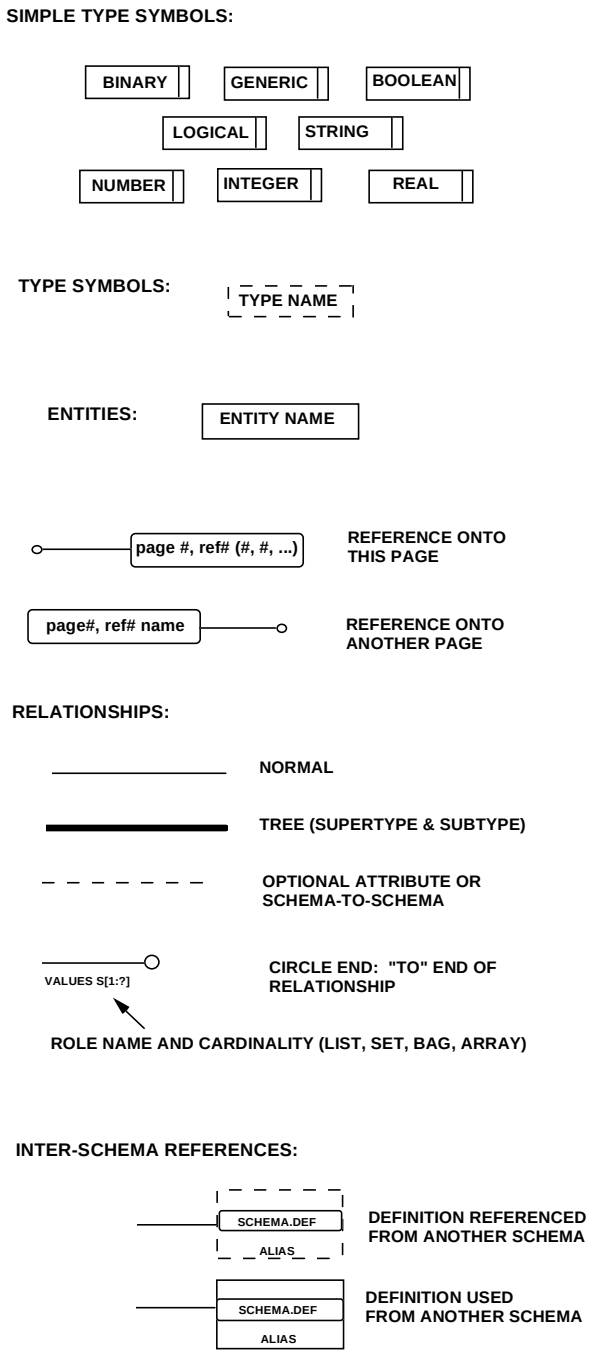

FIGURE 7. BASIC EXPRESS-G SYNTAX.

EXPRESS Model of Electronic Package Engineering

The initial entity of the EXPRESS model of electronic package engineering and an EXPRESS-G representation of that entity can be found in Figures 8 and 9, respectively. (The full model is many pages long.)

```
SCHEMA ELECTRONIC_PACKAGE_ENGINEERING;
ENTITY Microelectronic_Package_Product;
    ProductID: String;
    Description: String;
    PackageSubstrate: Package_Substrate;
    Design: Physical_Design;
    Materials: SET[1:?] OF Raw_Material;
    Components: SET[1:?] OF Electronic_Component;
    DesignedAccordingTo: SET OF Design_Standard;
    Functional_Test: Functional_Test_Results;
    Rework: Reworked_Package;
    Documentation: Documentation;
    Costs: Costs;
END_ENTITY;
…
END_SCHEMA;
```

FIGURE 8. EXPRESS MODEL OF MICROELECTRONIC PACKAGE PRODUCT.

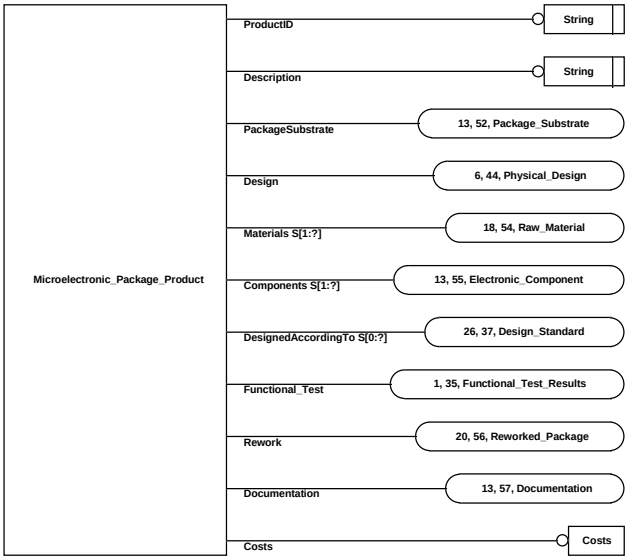

FIGURE 9. EXPRESS-G REPRESENTATION OF EXPRESS MODEL.

The EXPRESS model entities have many cross-references to other entities on other pages; hence, the figures are labeled with page numbers. Thus in Figure 9, there is one main entity, Microelectronic Package Product, having a number of attributes and other entities related to it. The Microelectronic Package Product entity has a ProductID (which is simply a String), a Package Substrate (another entity, found on another page of the EXPRESS-G Model), a set of Raw Materials, and so on.

The EXPRESS modeling performed in this project focused on a relatively high level of abstraction. A fully detailed information model for an electronic package was well beyond the scope of this project. An EXPRESS model of an electrical

schematic alone would be rather large, including all the symbology, components, component characteristics, locations on the drawing, etc. The STEP program has developed an Application Protocol (AP 210) for Printed Wiring Board design; the complete documentation for that EXPRESS model alone exceeds six hundred pages of definitions, text, and graphical symbology.

As mentioned before, the project team used software tools from STEP Tools, Inc. to develop the EXPRESS and EXPRESS-G models. STEP Tools, Inc. has also developed a set of tools (STEP-Developer) to implement persistent entity databases based on EXPRESS models. The model developed for this project was implemented in ROSE, the database management system provided through STEP-Developer. A ROSE database consists of C++ classes (incorporating data members and methods) which are manipulated by ROSE Library functions. Hence, a C++ program can include embedded ROSE code to access and manipulate data from the database.

RECOMMENDATIONS

To reap the potential benefits of the models reported here, these models should be extended and applied. Although the development of comprehensive models covering the breadth and depth of the electrical package engineering process was beyond the funding scope of the current project, such models should be developed. Any package engineering organization desiring to make use of this approach should also develop as-is models of their existing engineering processes and use the resulting models to critique their processes and evaluate alternative approaches.

The mapping of existing software tools and frameworks onto the models should be performed. Since this mapping would provide an immediate indication of limitations and problems in the areas of functional coverage, integration, and interoperability, organizations contemplating the licensing or development of automation software would have an improved basis for evaluating products and development plans. This information should aid tool vendors as well as tool users. The information should also be used to define specific integration and interoperability requirements needing to be addressed by tool developers along with detailed proposals for appropriate data format and protocol standards.

To date, the software tools for the engineering community have mostly been aiding tools, addressed to narrow portions of the problem and automating primarily well defined operations, such as simulation. However, the research community has been developing tools incorporating higher levels of capability, based on technology such as logic programming, knowledge based systems, and neural nets. Increasingly, these advanced tools are of necessity becoming users of the lower level tools (e.g., simulators) many of which were originally designed with a human user in mind and therefore biased, sometimes exclusively, to human-oriented modes of input and output. This development has led to the need for new methods and standards for tool invocation and tool-to-tool access of functionality. The lack of these methods and standards has already become a factor retarding advancement of automation tools for the engineering community. Research in these areas and resolution of these problems should be promoted. (The needs discussed here are not to be confused with those addressed by approaches such as Object Linking and Embedding (OLE)).

Even with an ideal software environment, nothing short of total automation could shield the user from much of the complexity inherent in the electronic package engineering process. It can be argued that complete automation would not be desirable even if it were attainable, that some mixture of automation and aiding that would keep a human in the loop would be preferable. Consequently, there may always be some burden on the engineer to understand, operate and orchestrate both the process activities and the tools. There will also be a need to train new users and possibly retrain previous users of the tools. Much of the operational burden and the need for special training can be substantially reduced by providing an intelligent software component to monitor the user's activities, answer high level questions, and provide guidance through the electronic package engineering process. This capability is within the reach of current software technology and could be implemented before many of the other advances recommended. The formal models advocated in this report would also provide much of the process-related information required to implement the capability. Therefore, we recommend the development of a prototype with this capability as a proof of concept.

SUMMARY

The goal of the project reported on in this document has been to develop information and activity models of the electronic package engineering process. The purpose of these models is to support analysis and improvement of this engineering process, especially through research and development in automation software and software integration. While the current models are of a generic nature, they provide examples and starting points for organizations to develop models specific to their own engineering processes. They also are immediately useful in evaluating the coverage, integration, and interoperability of existing automation software. Properly applied to process improvement and the development of appropriate automation tools, these models can lead to lowered cost and improved quality and performance of electronic packaging. The full models developed in the project can be found in [Cox and Jazbutis].

REFERENCES

Bravoco, R. R. and Yadav, S. B., "Requirement Definition Architecture --- An Overview," Computers in Industry, Vol. 6, pp. 237-251, 1985.

Bravoco, R. R. and Yadav, S. B., "A Methodology to Model the Functional Structure of an Organization," Computers in Industry, Vol. 6, pp. 345-361, 1985.

Cox, F. L. and Jazbutis, G. B., Process and Data Modeling of Electronic Package Engineering, Georgia Tech Research Institute, Atlanta, Georgia, April 1996.

Schenck, D. A. and Wilson, P. R., Information Modeling: The EXPRESS Way, Oxford University Press, New York, N.Y., 1994.

Mayer, R. J., et al., "Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report," Knowledge Based Systems, Inc., College Station, TX.

ISO CD 10303-210: Product Model Data Representation and Exchange - Part 210; Application Protocol: Printed Circuit Assembly Product Design Data N5-94; 1994.

ISO TC184/SC4/WG5 Part 11, EXPRESS Language Reference Manual, 1991.

STEP Tools, Inc., STEP-Developer Reference Manual, 1993.

Tummala, Rao R., and Rymaszewski, Eugene J. [ed.] Microelectronics Packaging Handbook, Van Nostrand Reinhold, New York, N.Y., 1989.

Yeh, Chao Pin, "An Integrated Information Framework for Multi-Disciplinary Printed Wiring Board Design," Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, 1990.