1989

METAPLEX: AN INTEGRATED ENVIRONMENT FOR ORGANIZATION AND INFORMATION SYSTEMS DEVELOPMENT

Minder Chen
George Mason University

Jar F. Nunamaker Jr.
University of Arizona

METAPLEX: AN INTEGRATED ENVIRONMENT FOR ORGANIZATION AND INFORMATION SYSTEMS DEVELOPMENT

Minder Chen
Department of Decision Science
George Mason University

Jay F. Nunamaker, Jr.
Department of Management Information Systems
University of Arizona

ABSTRACT

This paper presents an integrated environment, called MetaPlex, for organization and information systems development. The kernel of MetaPlex is a knowledge base management system which captures the semantic primitives of a domain at the meta level and uses these primitives to describe target systems. Three levels of abstraction are used in MetaPlex for representing knowledge: the axiomatic, median, and instance levels. The MetaPlex Language Definition System is used to name the object types in the domain of interest and to define the attributes, relations, and descriptions which can be used by these object types. The structural knowledge of the domain in general is thus captured at the median level. Knowledge of the domain captured at the median level is used by the MetaPlex Specification System to define a target system at the instance level. A rule-based inference engine is embedded in the MetaPlex environment as an intelligent assistant to help end users. The expertise of a designer can be codified into a rule set which can assist users in classifying an object, in decomposing a high level system component, or in clustering the detailed components at the lower level. Both top-down and bottom-up approaches for systems development are thus supported. A layered approach has been proposed to manage the dynamics of such a metasystem environment. An enterprise model has been developed to demonstrate the usage of MetaPlex and the integration of organization and information systems modeling. Directions for future research are also discussed.

1. THE INTEGRATION OF ORGANIZATION AND INFORMATION SYSTEMS DEVELOPMENT

Information systems (IS) are used to facilitate the decision making and communication processes within an organization and have recently been used by companies as strategic weapons against their competitors (Ives and Learmonth 1984). Information systems planning methods such as Critical Success Factor (CSF) (Rockart 1979) and Business Systems Planning (BSP) (IBM 1984) have begun to address the importance of business objectives to the determination of information systems requirements. Both BSP and CSF use interviews to involve managers in the information systems planning process. However, none of the models can adequately represent the couplings between business systems and information systems development. Only when business systems and information systems development are tightly connected can managers quickly respond to the changing business environment and take advantage of new IS technologies.

The lack of an integrated development environment and methods is one of the major causes of the missing link between organization (i.e., business systems) and information systems modeling. An integrated environment should allow users to specify the complicated linkages between business systems and information systems so that any changes in an information system can be reflected in its corresponding business counterpart, and the dynamics of the business environment can be propagated to the supporting information systems. We first review some existing computer-aided systems development approaches and discuss their advantages and disadvantages. Then we propose a software architecture for implementing an enterprise model to integrate organization and information systems development. Three major approaches are taken for the integration of systems development efforts: structured systems development methods, data dictionary systems, and metasystem approach.

- **Structured Systems Development Methods.** Methods in this category use a set of predefined terms to describe a target system. Efforts have also been made to extend them to cover a broader scope of the systems development life cycle. The advantage of this approach is its simplicity. The popularity of Structured Analysis Technique is due to its use of only four basic symbols (i.e., Data Flow, Data Store, External Entity, and Process) and its graphic representation (Gane and Sarson 1982). Jackson's Method (Jackson 1983) and SADT (Ross, Goodenough and Irvine 1977) also belong to this category. The disadvantage of tradi-

tional systems development methods is that their underlying models may not fit in with users' mental models of the real world. Although users can learn these methods quickly, they have to spend significant effort in transforming their views of a target system to a description constrained by the syntax of a certain method. Consequently, some real world meanings are either lost or distorted during the transformation process.

- **Data Dictionary Systems.** Automated tools can be integrated through a data dictionary system. Excelerator, using its XL Dictionary to integrate its Documentation, Analysis, Screens, and Graphics Tools, is an example (Patman 1986). Data Dictionary Systems have also been extended and used by information managers to manage information resources such as tasks, information flows, and relationships. The extended software is called Information Resource Dictionary System (Fife 1984). However, Data Dictionary Systems are still restricted in their capability to handle relations among objects.

- **Metasystem Approach.** The metasystem approach has the flexibility of allowing users to define their own terms to describe a target system (Kottemann and Konsynski 1984; Yamamoto 1981). The disadvantage of most existing metasystems comes from their poor interface design and the overhead costs. The syntactical complexities of their meta languages also make it difficult to define a new language.

The quality of the system under development is bounded by the available languages and tools as well as the way people use them (Lyytinen 1985). The broad scope and the rich semantics of developing a language to describe an enterprise make it difficult to use traditional structured methodologies and the data dictionary approach. Only the metasystem approach allows an organization to define languages for business and information systems development based on a basic model. The flexibility of the metasystem approach narrows the semantic gap between the specification tools and the application domains, thereby facilitating user learning and acceptance of the generated specification tools. We use the metasystem concept and a layered technique to remedy drawbacks of the metasystem approach. A generic knowledge representation scheme for describing a wide spectrum of systems in an organization is first presented. The kernel of MetaPlex, a layered approach to build a metasystem environment, and an example of using MetaPlex to build an enterprise model are discussed.

2. THE KNOWLEDGE REPRESENTATION IN METAPLEX

A system can be defined as a group of related components which interact with each other to achieve a high level objective of the system as a whole. Both organizations and information systems are instances of such a generic system. To describe an existing system or to design a new system, systems analysts or designers are mostly concerned with the system components and relationships among them. Various knowledge representation schemes have been studied to see whether they can be used for system specifications. A rule-based system can be used for capturing the designer's know-how on certain design decisions. A frame-based system can be used to represent general design schemata (Lubars and Harandi 1986). However, neither of these can explicitly represent the interrelationships of objects in a system. The knowledge representation scheme used in MetaPlex is based on a three-level abstraction of an object-oriented model: the axiomatic, median, and instance levels (Kottemann and Konsynski 1984). Figure 1 shows an example of this model.

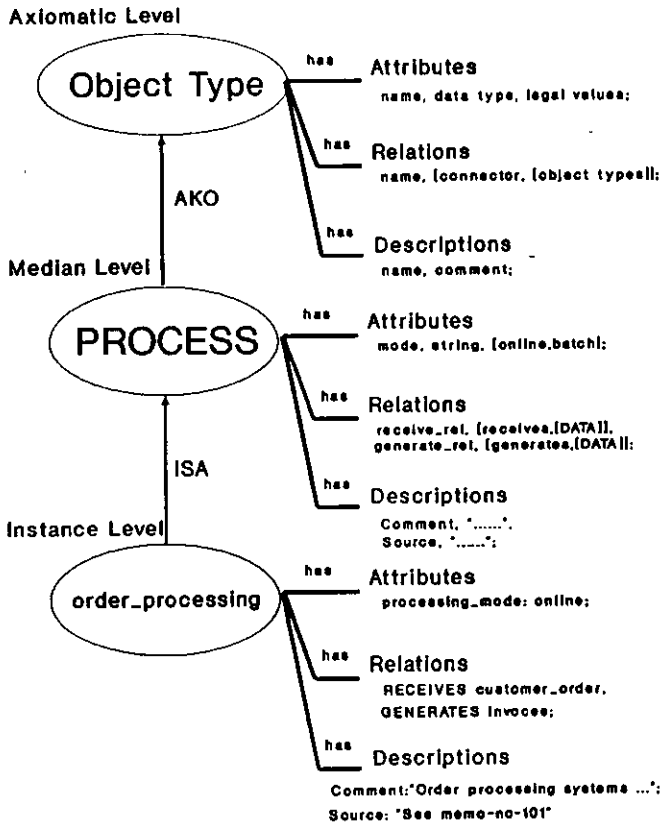

Figure 1. MetaPlex Knowledge Representation Model

Terms at the axiomatic level are built into the system. In MetaPlex, a domain can be defined only by object types which have attribute types, relation types, and description types. Attribute types have name, data type, and legal value. Relation types specify the interaction among object types. Description types include unstructured and procedural knowledge, as well as decision rules in text format.

At the median level, the object types in the domain of interest have to be identified along with the definition of attribute, relation, and description types. The structural knowledge of the domain in general is thus captured at the median level. Knowledge of the domain captured at the median level will be used to define a target system at the instance level.

2.1 The Cross Referencing in MetaPlex

In MetaPlex, a relation type is defined as two, or more than two, object types groups connected by binary connectors among them, as shown in Figure 2.

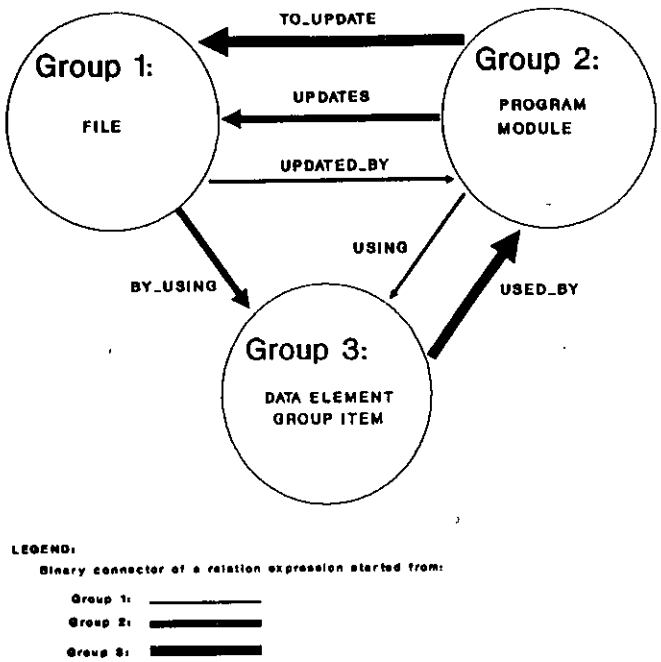

Figure 2. The Cross Referencing of a Relation Type

The language definition of the relation type in Figure 2 can be stated as one of the following three structured statements:

(a) [FILE] UPDATE_BY [PROGRAM,MODULE]
    USING [DATA ELEMENT, GROUP ITEM]
(b) [PROGRAM,MODULE] UPDATES [FILE]
    BY_USING [DATA ELEMENT,GROUP ITEM]
(c) [DATA ELEMENT,GROUP ITEM] USED_BY
    [PROGRAM,MODULE] TO_UPDATE [FILE]

The complementary ways of describing the same relation that exists in different object type groups have to be defined in the language definition so that the system can automatically do the cross referencing on the target system specification. By bringing all relevant information about an object together in one report, the cross reference capability reduces the effort required of a user to describe a target system and facilitates the user's understanding of the target system specification. For example, once a user has entered the following relation:

payroll_processing_program
    UPDATES payroll_master_file
    BY_USING new_employee_information

MetaPlex can figure out its complementary relations as:

(a) payroll_master_file
        UPDATE_BY payroll_processing_program
        BY_USING new_employee_information

(b) new_employee_information
        USED_BY payroll_processing_program
        TO_UPDATE payroll_master_file

The MetaPlex system will ask users about the object type of a new object entered through a relation and check the consistency of object types in existing objects. If more than one object type is possible for a new object, the system will prompt the user with all possible object types in a selection menu. If only one object type corresponds to a newly entered object, MetaPlex will automatically assign the object type to it. As an example, using the newly entered relation above and assuming that both **payroll_processing_program** and **payroll_master_file** are new objects, the system will ask the user to choose from **PROGRAM** and **MODULE** the object type of **payroll_processing_program** while **FILE** will be automatically assigned to **payroll_master_file** as its object type.

The same object type may appear in a relation type more than once. For example:

(a) [PROCESS] COMES AFTER [PROCESS]
(b) [GROUP ITEM] CONSISTS OF [GROUP ITEM, DATA ELEMENT]

The first relation defines the time sequence of two processes. The second relation describes the decomposition relation of objects, which may be either direct or indirect recursive. Relations of object types have been classified into various perspectives, called system aspects. A user is able to choose a system aspect so as to concentrate only on certain relations at a particular time.

Currently, MetaPlex uses "structure" in the Prolog language to represent both the definitions of MetaPlex languages and their target system specifications. "LIST" is simply a special "structure" in Prolog (Clark and Mellish 1984). The declarative nature of this internal data structure makes the representation and manipulation of languages and target system specifications much easier.

2.2 Representing Abstraction in MetaPlex

Researchers have identified three major abstraction mechanisms for describing a target system: Classification, Generalization/Specialization, and Aggregation (Gibbs 1985). The equivalent representations of these abstraction mechanisms in MetaPlex, discussed below, demonstrate the

expressive power of the MetaPlex knowledge representation scheme.

1. **Classification.** The "object type" and "object" in MetaPlex are equivalent to "class" and "instance." The properties and relationships defined for an object type are used to elicit information about objects of this type in a target system. For most applications, two levels of classification are sufficient (Mylopoulos, Berstein and Wong 1980).

2. **Generalization/Specialization.** The class hierarchy in an object-oriented system can be represented by using an "AKO" relation among object types. For example, we can define "REPORT" IS A KIND OF "DATA" and "MONTHLY REPORT" IS A KIND OF "REPORT." Property inheritance along the class hierarchy is handled by an inference engine.

3. **Aggregation.** There are two types of aggregation: Cartesian aggregation and cover aggregation. In MetaPlex, Cartesian aggregation means that an object type is an aggregation of its attribute and description types. The cover aggregation can be specified by using a decomposition relation in the following format:

   [OBJECT TYPE] HAS PARTS [OBJECT TYPE]
   [OBJECT TYPE] IS PART OF [OBJECT TYPE]

MetaPlex does not impose constraints on how many objects can be involved in a relation. Users can describe one-to-one (linear structure), one-to-many (tree structure), and many-to-many (network structure) relations. Any complicated relation can be easily represented in MetaPlex.

3. The Architecture of the MetaPlex Kernel

The design goal for MetaPlex is to develop a simple, but flexible, computer-aided specification tool that can be applied to various domains. The simplicity of MetaPlex is achieved through an interactive menu-driven user interface and a graphic representation of a target system specification. Procedural knowledge can be captured in the narrative descriptions associated with objects. A specialized inference mechanism can be developed to handle various procedural knowledge and to interpret special relations.

The architecture of the MetaPlex Kernel is shown in Figure 3. It has two subsystems: the MetaPlex Language Definition System and the MetaPlex Specification System.

The kernel of MetaPlex can be used as a knowledge base management system to manage specification languages, design expertise, and target system specifications. While other metasystems use compilation (Yamamoto 1981; Demetrovics, Knuth and Rado 1982), MetaPlex uses an interpretation approach for language definition and target system specification. The interpretation approach makes it much easier to develop specification languages. Eventually, users will be able to develop languages of their own without any help from system administrators. Detailed design of the MetaPlex Language Definition System and the MetaPlex Specification System as well as the procedure of using them are described in the remainder of this section.

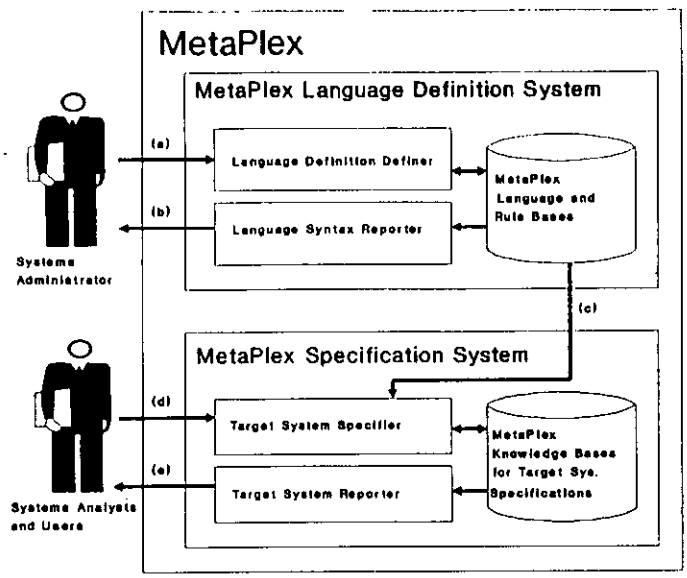

Figure 3. The Architecture of the MetaPlex Kernel

3.1 The MetaPlex Language Definition System

The MetaPlex Language Definition System allows systems administrators to define and maintain object types and their associated properties, define and maintain relation types among object types, manage the language database, and generate the Language Syntax Report.

The Language Syntax Report is used by systems administrators and systems analysts. The Language Syntax Report prints out all of the attributes, relations, and descriptions which will possibly be used by an object of a certain type. Figure 4 is a partial listing of a Language Syntax Report. To support multiple user views, MetaPlex allows the systems administrator to create sub-languages from a MetaPlex language so that users can use a sub-language to interact with the knowledge base system.

3.2 The MetaPlex Specification System

The MetaPlex Specification System (MPSS) is used to define and analyze the specification of a target system. To use the Target System Specifier, users have to load a modeling language at the beginning of a session. They can then start to specify the objects in the target system by

```
**************************************************************
*    Language Syntax Report                                  *
*    Project Name: PLEXSYS PROJECT                           *
*    Date: 1987-5-21      Time: 11:46:38                     *
**************************************************************

Object Type Name: INPUT
Comment Is:
Any input is a data carrier which is received and used by the proposed
system. An input is created externally to the system by one or more
interfaces and enters the system from outside.
;

Attributes are:
    ARRIVAL_TYPE(STRING):
        monthly, bi-weekly, weekly, daily, randomly;

Relations are:
    Systems Aspect: SYSTEM I/O FLOW
    Relation Name: INPUT <-> PROCESS
    RECEIVED BY: PROCESS;

    Systems Aspect: SYSTEM I/O FLOW
    Relation name: INTERFACE <-> PROCESS
    GENERATED BY: INTERFACE;

    Systems Aspect: SYSTEM STRUCTURE
    Relation Name: INPUT STRUCTURE
    IS PART OF: INPUT;

    System Aspect: SYSTEM STRUCTURE
    Relation Name: INPUT STRUCTURE
    HAS SUBPARTS: INPUT;

Descriptions are:
    LAYOUT:
        A layout of an input form.      ;
    DESCRIPTION:
        A description of the INPUT.     ;

            :
            :

**************************************************************
*    The End of the Language Syntax Report                   *
**************************************************************
```

Figure 4. MetaPlex Language Syntax Report

associating new objects with object types defined at the median level. Once the object type of an object has been defined, its related attribute, relation, and description types will be used to elicit the requirements of the target system.

The Target System Reporter supports general reporting and on-line query utilities. Reports in various formats (i.e., in text, table, or graphics) can be generated. Two major reporting utilities are Structure Diagram Browser and Formatted Specification Statements. The Structure Diagram Browser utility allows users choose an object from a menu, select existing relations associated with this object, and draw a structure diagram by using the object as the root of a tree (see Figure 5).

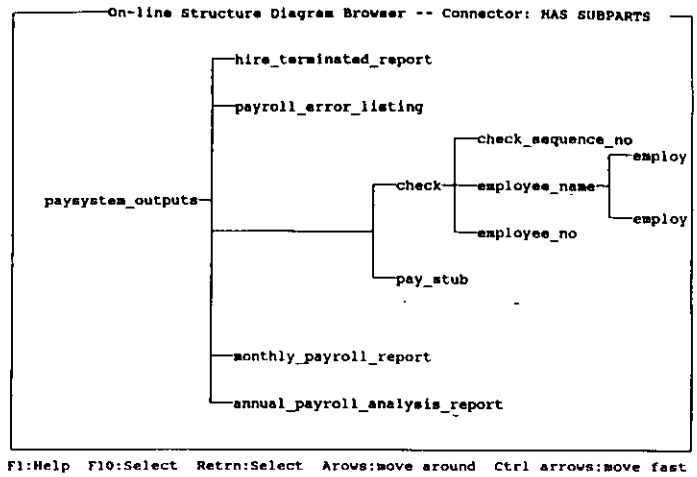

Figure 5. MetaPlex Structure Diagram Browser

The Structure Diagram is a virtual tree which can be drawn in one of the four directions. Using the same facility, users can browse through the objects in the tree to examine their attributes and descriptions or look at their relations with other objects. An object name will be suffixed by "(LOOP)" if it appears more than once in the Structure Diagram. The Formatted Specification Statements Report is a complete specification of the target system database in which all of the relations have been cross-referenced, as shown in Figure 6.

A callable rule-based system, functioning as an intelligent assistant, is embedded in the MetaPlex environment to help end users. The expertise of a designer can be codified into a rule set which can assist users in typing an object (i.e., identifying the object type of an object), in decomposing a high level system component, or in clustering the detailed components at a lower level (Karimi 1987). Both top-down and bottom-up approaches for systems development are thus supported. The completeness and consistency constraints of the target system specification are designed to be specified in rule format and enforced by the rule-based system (Kang 1982; Vickery, Brooks and Vickery 1986).

3.3 A Process of Using MetaPlex

A logical process of using MetaPlex, as labeled in Figure 3, is described in this subsection.

a. A systems administrator uses the Language Syntax Definer to define a language conforming with the MetaPlex Language Syntax.

b. The syntax and descriptions of the defined language can be generated from the system as a user manual for the systems administrator and systems analysts to use.

```
**********************************************************************
*    Formatted Specification Statements                              *
*    Project Name: PLEXSYS PROJECT                                   *
*    Date: 1987-5-19      Time: 3:38:51                              *
**********************************************************************

Object Name: employees
Object Type: INTERFACE
Attributes Are:

Relations Are:
     Systems Aspect: SYSTEM I/O FLOW
     Relation Name: OUTPUT <-> INTERFACE
     RECEIVES: pay_statement;

     Systems Aspect: SYSTEM I/O FLOW
     Relation Name: INTERFACE <-> INPUT
     GENERATES: time_card;

     Systems Aspect: SYSTEM STRUCTURE
     Relation Name: INTERFACE STRUCTURE
     IS PART OF: departments_employees;

Descriptions Are:

**********************************************************************
Object Name: time_card
Object Type: INPUT
Attributes Are:
     ARRIVAL_TYPE: weekly;

Relations Are:
     Systems Aspect: SYSTEM I/O FLOW
     Relation Name: INTERFACE <-> INTERFACE
     GENERATED BY: employees;

Descriptions Are:

     LAYOUT:
          Name:_____________________________
          Employee Number:_____________________
          Hours:_____________________________
          ;
     :
     :

**********************************************************************
*    END OF Formatted Specification Statements Report                *
**********************************************************************
```

Figure 6. Metaplex Formatted Specification Statements

c. After having loaded a language previously defined by the systems administrator, a systems analyst can use the MetaPlex Specification System to define a target system.

d. Systems analysts can use the Target System Specifier to enter new objects of the target system and identify their object types. Once the object type of an object has been defined, the Target System Specifier prompts systems analysts for all the possible attributes, descriptions, and relations.

e. Systems analysts can use the Target System Reporter, which includes on-line query and off-line printing utilities, to generate various reports from the target system database.

Under the MetaPlex environment, the specification of requirements is an incremental development process. Detailed information can be added to the system and changes can be made. Systems administrators can add new object types as well as relation, attribute, and description types into an existing language without changing related existing target systems specifications. This capability allows the language defined by MetaPlex to grow with the changing needs of applications.

4. A LAYERED APPROACH TO MANAGING THE DYNAMICS OF A METASYSTEM ENVIRONMENT

A metasystem environment can be built on top of the MetaPlex kernel. A layered approach is proposed to manage the dynamics of the metasystem environment (see Figure 7).

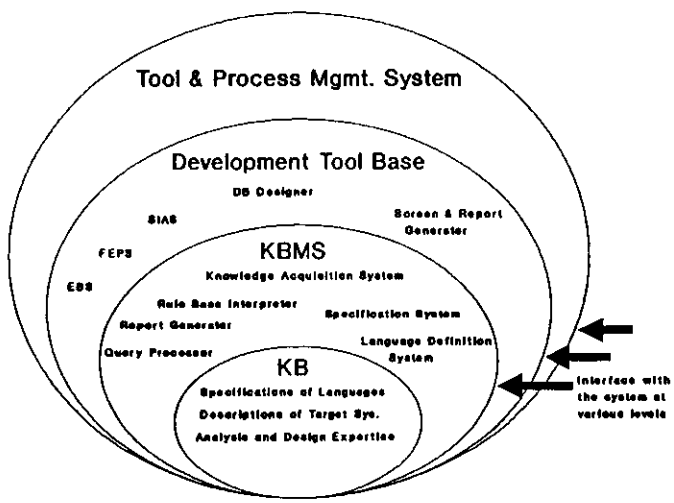

Figure 7. Building a metasystem Environment: A Layered Approach

The functionalities of each layer from inside out are outlined as follows:

1. **MetaPlex Knowledge Base.** The MetaPlex Knowledge Base at the center serves as a centralized repository for language definition and target system specifications, as well as for analysis and design expertise.

2. **MetaPlex Knowledge Base Management System.** The MetaPlex KBMS manages knowledge about the specification language, analysis and design expertise, and target systems descriptions. MetaPlex supports general utilities and tools for managing knowledge acquisition, report generation, and query processing.

3. **Development Tool Base.** A spectrum of tools for organization modeling, such as FEPS and SIAS, and

for information systems analysis and design, such as DB Designer and Screen and Report Designer (PLEXSYS 1987), is developed to customize user interface. Tools in the Development Tool Base can be linked with the knowledge base access procedure under the same programming environment or can communicate with the MetaPlex KBMS through the knowledge base access commands in a command file.

4. **Tool and Process Management System.** The Tool and Process Management System at the outermost layer includes the Tool Management System and the Process Management System. The Tool Management System can select a set of tools from the Tool Base for a specific organization, problem, or process. The input and output relationships, functionalities, and compatibilities among tools can be represented in MetaPlex. A tool selection expert, such as an Information Center Expert (Heltne et al. 1987), can suggest to users which tools could be used for various problems. The Process Management System can help users by suggesting procedures for a planning or decision making session, selecting a series of tools to facilitate a development process, controlling the progress of systems development processes, and documenting the results and responsibilities of processes.

The kernel of MetaPlex can be used as a stand-alone tool for building the knowledge base directly so that users are able to adopt the system without having an existing tool base. The Development Tool Base Layer can be developed to customize the user interface for specific applications. Eventually, the whole development process is under the control of the Tool and Process Management System so that users who have no prior knowledge about the knowledge base system still feel comfortable using it.

5. A CASE STUDY OF METAPLEX: BUILDING AN ENTERPRISE MODEL

The current thrust of the PLEXSYS project (Konsynski et al. 1984) is to build an integrated environment for organization and information modeling, i.e., enterprise modeling. The enterprise model is a snapshot of the organization as it is or a design of an organization in the future. A model of what comprises an enterprise is described by a language, called the Enterprise Description Language, through the MetaPlex Language Definition System. The process of building and updating the enterprise model for an organization is facilitated by the MetaPlex Specification System and Development Tool Base, which is interfaced with the knowledge base management system. In the following section, we will discuss how we can define a language for an enterprise to model its external environment and internal operations.

Figure 8 presents a proposed enterprise model. (Note that not all of the relationships discussed below are included in this figure and most of attributes and descriptions for object types are not discussed in this paper.)

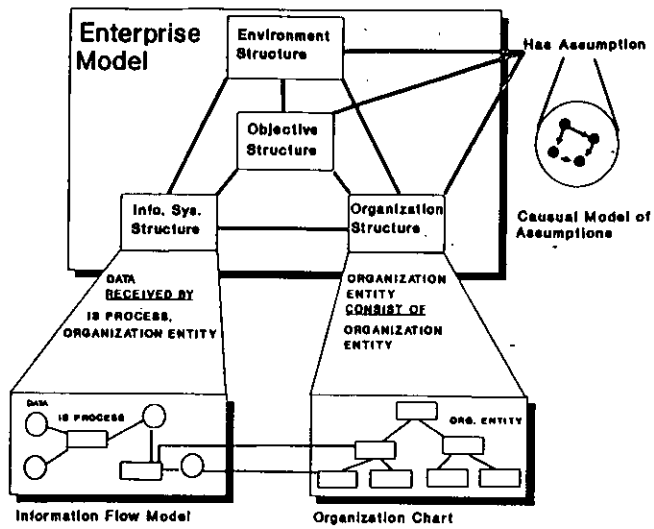

Figure 8. An Enterprise Model

An enterprise, from a static point of view, consists of the organization structure, objective structure, and information systems structure within the organization boundaries, as well as the environment structure outside the boundaries. From a dynamics viewpoint, these four components will interact. For example, the competitive forces in the environment constrain the objective setting and policy formulation of an organization. The objective structure of the organization will guide the functioning of organization entities and business processes, and the design and development of information systems. Information systems support the business processes performed by organization entities, generate performance reports which are used by managers to measure the achievement of objectives, or are even used directly to achieve certain objectives. The four major components in the enterprise model and the relations among them are discussed here.

Organization Structure. An organization structure has two major substructures: organization entity structure and business process structure. Organization entity, as a basic element in the organization entity structure, is defined as a strategic business unit or an organization unit. A user can draw an organization chart by giving the relation:

[ORGANIZATION ENTITY]
CONSISTS OF [ORGANIZATION ENTITY]

Business process structure is a breakdown of the functions and activities of an organization. These two substructures are linked together by assigning responsibilities of business processes to organization entities through the following relation:

[ORGANIZATION ENTITY]
IS RESPONSIBLE FOR [BUSINESS PROCESS]

Environment Structure. The environment of an organization consists of external entities which influence the objectives, policies, and business process of an organization. Examples of these external forces are customers, suppliers, competitors within the industry, potential entrants, and substitute products (Porter 1983). At the macro level, environment also includes government regulations, technology innovation, and social and political situations. EXTERNAL ENTITY as an object type is used to represent those environmental entities. The objectives formulated by an organization result from balancing the claims of stakeholders (i.e., external entities and organization entities) from the internal organization and external environment (Mason and Mitroff 1981; Freeman 1983). The strategic assumption surfacing technique is used to explicate assumptions of stakeholders and to construct their relationships with the objectives of the organization. The relation can be defined as

[EXTERNAL ENTITY, ORGANIZATION ENTITY]
HOLDS [ASSUMPTION]
RELATED TO [OBJECTIVE]

Assumptions held by both organization entities and external entities can be linked together to form a casual model for qualitative modeling (Bobrow 1985). An example of a relation type between assumptions is

(a) [ASSUMPTION] ENFORCES [ASSUMPTIONS]
(b) [ASSUMPTION] HOLD AGAINST [ASSUMPTION]

The path analysis method in casual modeling (Paradice and Courtney 1987) can be applied to study the interactions among assumptions that can help us to understand the dynamics of an organization and its environment.

Objective Structure. The objectives of an organization form a hierarchical or a network structure. Objectives at the highest level, called missions, are answers to questions such as "What business are we in," "Who are our customers," and "What is the direction of the company in the future?" Missions are used as guidelines for goal setting. Goals can be refined to objectives which are preferably defined by some measurable factors within a shorter time frame (Rowe et al. 1985). The authors have adopted a goal-driven view of an organization in which all the objectives involve capitalizing on the strengths and avoiding the weaknesses of the organization, as well as balancing the impacts of competing forces in the environment. Objective structure will guide the business processes and information systems of the organization at their respective levels, while achieving the success of objectives will be assigned as responsibilities to organization entities.

Ansoff (1956) argues that organizations should use objectives as their tools for management and that there should be tighter connections between the objectives of an organization and its information systems. Managerial functions, such as planning and controlling, are the processes that define and monitor organizational objectives at every level, while coordination and communication are mechanisms to achieve objectives through business processes. A language can be defined to capture relations, such as what reports are used to measure the performance of achieving an objective, what information systems and business functions are used to support or carry out objectives, what organization entities are responsible for the implementation of objectives, etc.

Information System Structure. Data and information system (IS) processes are two main components in an information system structure (Freeman and Wasserman 1984). Object type **DATA** is used instead of "input" and "output" because "input" and "output" are relative terms. Attributes being defined as "Data Type" can be used to distinguish types of data: REPORT, FORM, FILE, DATA BASE. Decomposition relations can be defined for both **DATA** and **IS PROCESS** to describe their substructures. The information flow within an organization and across its boundaries is described by the following relations:

(a) [IS PROCESS, ORGANIZATION, ENTITY, EXTERNAL ENTITY, BUSINESS PROCESS] GENERATES [DATA]
(b) [IS PROCESS, ORGANIZATION ENTITY, EXTERNAL ENTITY, BUSINESS PROCESS] RECEIVES [DATA]

A information flow model can be drawn from these relations.

A complete version of this enterprise model is still under development. The current discussion is presented to demonstrate how to use MetaPlex to define a language for organization and information systems modeling. In order to customize the development of a language for an enterprise, the authors have developed a procedure for defining an enterprise language in conjunction with the PLEXSYS Planning Tool (PLEXSYS 1987). A task force formed by managers and information systems analysts can use the following procedures to develop its own enterprise description language.

1. **Defining Object Types.** An Electronic Brain Storming (EBS) session held for a group of high level managers will be posed with the following question: "What are the most important object types (entities) in the enterprise?" The participants in the session will use the Issue Analysis (IA) tool to identify and consolidate EBS comments and to generate a list of essential object types.

2. **Defining Relations Among Object Types.** After all of the important concepts and entities have been identified, questions such as "What are the most important relationships among the following object types?" will be posed in the next round of EBS and issue analysis sessions. A list of relations among object types will be generated and ranked as a result of this process.

Attributes and descriptions also can be identified by using the same process.

3. **Defining and Revising the Language.** The MetaPlex Language Definition System will be used to define results from the process presented above as a language and generate a Language Syntax Report for the management team to review. The language will be revised in response to comments by managers and users.

A study has shown that users of the PLEXSYS Planning Tool had a high level of satisfaction with the outcome of their planning sessions and the process of achieving those outcomes (Nunamaker, Applegate and Konsynski 1987). The involvement of top managers is expected to increase the acceptance of the final description language and ensure that important high level concepts within the organization will be included in an enterprise description language. A set of idea generation, analysis, and design tools will then be developed according to the planning methods chosen by the managers. With the help of the MetaPlex Knowledge Base System, we foresee being able to manage the complicated relations of an enterprise so as to improve its business and information systems development.

6. FUTURE RESEARCH AND CONCLUSIONS

Besides the Enterprise Description Language, we have developed languages for Business Systems Planning, which is an information systems planning method (IBM 1984); a subset of the Problem Statement Language in PSL/PSA, which is a computer-aided documentation and analysis tool for information systems (Teichroew and Hershey 1977); and a genealogy record keeping system. MetaPlex has demonstrated its applicability to various application domains. However, we have found the following improvements are needed to enhance the capability and usability of MetaPlex.

1. **Enhance the Graphic Interface of MetaPlex.** Current graphic capabilities in MetaPlex are limited to character based graphics for on-line query of relations among objects. Defining an icon for each object type in the MetaPlex language would enable systems users to directly manipulate graphic icons to specify system requirements and generate customized output from a target system database. An object-oriented approach for the graphic interface, such as the GROW System (Barth 1986), will be used for the future graphic interface enhancement.

2. **Improve the Database Performance.** Specification of a complicated system could have a large number of objects and relations. The performance of updating and retrieving data in the knowledge base will become critical for an interactive operation. Low level file management routines have to be developed so that the knowledge base can be accessed and maintained efficiently.

3. **Use MetaPlex as a Hypertext System.** MetaPlex is capable of handling complicated relations and can represent attributes of an object by various media, such as text, graphic, or slide show (e.g., by using callable graphics software such as PC STORYBOARD). Managers can use this facility to organize their ideas and present them in textual or graphic formats. The browsing facility (Structure Diagram Browser) can help users go through a net of linked thoughts by means of a group discussion tool such as that proposed by Johansen (1987).

4. **Choose a Better Programming Environment for Implementation.** Currently MetaPlex is implemented in Turbo Prolog (Borland 1986) on a personal computer. The Prolog language has been used for the current implementation because it has been demonstrated to be effective for knowledge representation (Bowen 1985), implementing an objected-oriented approach (Shapiro and Takeuchi 1983), and software specification (Leung and Choo 1985). However, currently available Prolog systems provide little support for object-oriented programming. To take advantage of the built-in object classification hierarchy and property inheritance mechanism in object-oriented languages and tools, Smalltalk, ACTOR, or KEE, are now considered to be potential candidates for MetaPlex implementation in the future. For an architecture and design for an integrated development environment based on an active, object-oriented approach, readers may refer to Chen, Nunamaker, and Konsynski 1987.

Future success of information systems in an organization will depend on how information systems development can be coordinated with business systems. In this paper we have presented a metasystem approach which allows us to develop a software environment for integrating organization and information systems modeling. The usability of MetaPlex, compared with other manual and automated tools, is still open to empirical study. Possible findings will suggest the direction further design and implementation should take.

7. ACKNOWLEDGEMENTS

This Research is supported in part by the National Science Foundation, the United States Army, and IBM.

8. REFERENCES

Ansoff, H. I. *Corporate Strategy.* New York: McGraw-Hill Book Company, 1965.

Barth, P. S. "An Object-Oriented Approach to Graphical Interfaces." *ACM Transactions on Graphics*, Volume 5, Number 2, April 1986, pp. 142-147.

Bobrow, D. G., Editor. *Reasoning about Physical Systems*. Cambridge, Massachusetts: MIT Press, 1985.

Borland International Inc. *Turbo Prolog: Owner's Handbook*. 1986.

Bowen, K. A. "Meta-Level Programming and Knowledge Representation." *New Generation Computing*, Volume 3, 1985, pp. 359-383.

Chen, M.; Nunamaker, J. F., Jr.; and Konsynski, B. R. "PLEXACT: An Architecture and Design of a Knowledge-Based System for Information Systems Development." *Computer Personnel Journal*, August, 1987.

Clark, K. L., and Mellish, C. S. *Programming in Prolog*. Second Edition, Springer-Verlag, 1984.

Demetrovics, J.; Knuth, E.; and Rado, P. "Specification Meta System." *IEEE Computer*, May 1982, pp. 29-35.

Fife, D. W. "The Dictionary Becomes a Tool for System Management." In E. A. Unger, P. S. Fisher, and J. Slonim, Editors, *Advances in Database Management*, Wiley HeyDen, Limited, Volume 2, 1984, pp. 101-117.

Freeman, P., and Wasserman, A. I., Editors. *Tutorial on Software Design Techniques*. Fourth Edition, IEEE Computer Society Press, 1984.

Freeman, R. E. "Strategic Management: A Stakeholder Approach." In *Advances in Strategic Management*, JAI Press Inc., Volume 1, 1983, pp. 31-60.

Gane, C., and Sarson, T. "Structured Methodology: What We Have Learned?" In J. D. Couger, M. A. Colter, and R. W., Knapp, Editors, *Advanced Systems Development/Feasibility Techniques*, Wiley and Sons, 1983.

Gibbs, S. J. "Conceptual Modeling and Office Information Systems." In D. Tsichritzis, Editor, *Office Automation*, Springer-Verlag, 1985, pp. 193-225.

Heltne, M. M.; Vinze, A. S.; Konsynski, B. J.; and Nunamaker, J. F., Jr. "A Consultation System for Information Center Resource Allocation." In M. A. Elias, Editor, *Proceedings of the 1987 SIGBDP-SIGCPR Conference*, 1987.

IBM, *Business Systems Planning*. Fourth Edition, GE20-0527-4, July 1984.

Ives, B., and Learmonth, G. P. "The Information System as a Competitive Weapon." *Communication of ACM*, Volume 27, Number 12, December 1984, pp. 1193-1201.

Jackson, M. *System Development*. Prentice-Hall International, Inc., 1983.

Johansen, R. "User Approaches to Computer-Supported Teams." In *Technological Support for Group Collaboration: The 1987 NYU Symposium*, New York University, May 1987, pp. 29-62.

Kang, K. C. *An Approach for Supporting Systems Development Methodologies for Developing a Complete and Consistent System Specification*. Dissertation, The University of Michigan, 1982.

Karimi, J. "An Automated Software Design Methodology Using CAPO." *Journal of Management Information Systems*, Volume 3, Number 3, Winter 1986-1987, pp. 71-100.

Konsynski, B. R.; Kottemann, J. E.; Nunamaker, J. F., Jr.; and Stott, J. W. "PLEXSYS-84: An Integrated Development Environment for Information Systems." *Journal of Management Information Systems*, Volume 1, Number 3, Winter 1984, pp. 64-104.

Kottemann, J. E., and Konsynski, B. R. "Dynamic Metasystem for Information Systems Development." In *Proceedings of the Fifth International Conference on Information Systems*, pp. 187-204.

Leung, C. H. C., and Choo, Q. H. "A Knowledge-base for Effective Software Specification and Maintenance." In *The Third International Workshop on Software Specification and Design*, 1985, pp. 63-66.

Lubars, M. S., and Harandi, M. T. "Intelligent Support for Software Specification and Design." *IEEE Expert*, Winter 1986, pp. 33-41.

Lyytinen, K. J. "Implications of Theories of Language for Information Systems." *MIS Quarterly*, Volume 9, Number 1, March 1985, pp. 61-74.

Mason, R. O., and Mitroff, I. *Challenging Strategic Planning Assumptions*. John Wiley and Sons, 1981.

Mylopoulos, J.; Berstein, P. A.; and Wong, H. K. T. "TAXIS: A Language Facility for Designing Database-Intensive Applications." *ACM Transactions on Database Systems*, Volume 5, Number 2, June 1980, pp. 186-207.

Nunamaker, J. F., Jr.; Applegate, L. M.; and Konsynski, B. R. "Facilitating Group Creativity: Experience with a Group Decision Support System." *Journal of Management Information Systems*, Volume 3, Number 4, Spring 1987, pp. 1-19.

Paradice, D. B., and Courtney, J. F., Jr. "Casual and Non-casual Relationships and Dynamic Model Construction in a Managerial Advisory System." *Journal of Management*

Information Systems, Volume 3, Number 4, Spring 1987, pp. 20-38.

Patman, B. "An Introduction to Systems Design Techniques." In D. Martland, S. Holloway, and L. Bhabuta, Editors, *Fourth Generation Languages and Application Generators*, The Technical Press, 1986, pp. 112-137.

PLEXSYS Planning Tools: User's Guide. Management Information Systems Department, The University of Arizona, 1987.

Porter, M. E. *Competitive Strategy: Techniques for Analyzing Industries and Competitors*, The Free Press, 1980.

Rockart, J. F. "Chief Executives Define Their Own Data Needs." *Harvard Business Review*, Volume 57, Number 2, March/April 1979, pp. 81-93.

Ross, D. T.; Goodenough, J. B., and Irvine, C. A. "Software Engineering: Process, Principles, and Goals." *IEEE Computer*, Volume 8, Number 5, May 1977, pp. 17-27.

Rowe, A. J.; Mason, R. O.; Dickel, K. E.; and Sparks, J. E. *Strategic Management and Business Policy: A Methodological Approach*. Second Edition, Addison-Wesley Publishing Company, Inc., 1985.

Shapiro, E., and Takeuchi, A. "Object-Oriented Programming in Concurrent Prolog." *New Generation Computing*, Volume 1, Number 1, 1983, pp. 25-48.

Teichroew, D., and Hershey, E. A. "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems." *IEEE Transactions on Software Engineering*, Volume SE-3, Number 1, January 1977, pp. 41-48.

Vickery, A.; Brooks, H. M.; and Vickery, B. C. "An Expert System for Referral: The PLEXUS Project." In R. Davies, Editor, *Intelligent Information Systems: Progress and Prospects*, Ellis Horwood Limited, 1986, pp. 154-183.

Yamamoto, Y. *An Approach to the Generation of Software Life Cycle Support Systems*. Dissertation, The University of Michigan, 1981.