1984

Dynamic Metasystems for Information Systems Development

Jeffrey E. Kottemann
University of Hawaii

Benn R. Konsynski
University of Arizona

Dynamic Metasystems for Information Systems Development

Jeffrey E. Kottemann
Department of Decision Sciences
University of Hawaii

Benn R. Konsynski
Department of Management Information Systems
University of Arizona

ABSTRACT

Dynamics in the use of metasystems in the development of information systems is discussed. An axiomatic level of specification is used to allow dynamic specification of "median" level metasystems which are, in turn, used in information systems specification, analysis and design. Existing metasystems are reviewed and principles for metasystem evaluation are considered. The implementation and use of dynamic metasystems in the Plexsys system is overviewed. The Plexsys system implements generalized integrity analysis at all levels of logic and mechanisms to insure the mutual integrity of these levels over time.

Introduction

A science is a well made language.
—Condillac

Computer-aided environments are evolving to facilitate the specification and development of large-scale information systems. Tools to support enterprise analysis, logical data and process modeling, database design, process organization, automatic code generation, and other design activities exhibit a variety of models, semantics, and terminology. A degree of dynamics must be introduced if the design support tools are to be effective. These dynamics are fundamentally important as both language definitions and target models change over time. That is, as more is learned about the organization and about the development process itself, the development environment must support the modification of language and target model definitions such that the models are *internally* and *mutually* complete and consistent.

Conceptual underpinnings, as well as the structure and function of metasystems, are discussed in this paper. A metasystem framework of three basic definitional levels is developed. Requirements for an effective metasystem are outlined, including succinctness, dynamism, scope, and granularity. Three metasystems used in information systems specifications—SEM, SDLA, and SDS—are analyzed. The emphasis of the analysis is an assessment of the degree to which the metasystem can be a guarantor of the integrity of models. The integrity of a model concerns its *semantic completeness.* An alternate meta-paradigm is proposed. The paradigm draws on the relationship of the metasystems concept to semantics and knowledge representation in linquistics and artificial intelligence. The meta approach and system presented allows for the generalization of a set of integrity rules for language specifications and target system descriptions. The implementation, which provides for dynamism of the overall three-tier model, is discussed in the final section.

The Metasystem Concept

In describing information systems, a large set of often disjoint terminology is used among development settings. In many cases, several terms are used to name a given term or concept. For example, "record," "group," "relation," and "data structure" have all been used to name a conceptually analogous term.

One major drawback of many computer-aided methodologies is that the predefined terms used in the methodology may not be the same as the terms used by target system developers in any given setting. This drawback leads to one of two outcomes; namely, the computer-aided methodology will not be adopted, or it is adopted with the accompanying cost of reorienting all individuals involved in systems development. In the second outcome, extensive training of developers with respect to understanding the "packaged" system view and becoming fluent in the methodology's terms is required.

Despite the existence of differences among development settings, these differences can be isolated, through the process of abstraction. Through this abstraction process a conceptual, or axiomatic model is attained which can be used to define modeling forms for relatively disparate development settings. The result is a computer-aided methodology that is tailorable to any of a set of development environments.

UNDERLYING METASYSTEM PRINCIPLES

A metasystem view consists of definitional levels. The three metasystem levels of Figure 1 have been named with the adjectives "axiomatic," "axiomatamedian," and "instantial." The definition of these, given in Webster's Third New International Dictionary, are:

1. *Axiom:* a proposition, principle, rule of maxim that has found general acceptance or is thought worthy thereof whether by virtue of a claim to intrinsic merit or on the basis of an appeal to self-evidence.

2. *Axiomatamedia:* the general principles that are above simple empirical laws yet inferior to the highest generalizations or those that are taken to be fundamental.

3. *Instantial:* reference to any particular person, thing, or situation.

For the sake of parsimony, *median* will be used in place of axiomatamedian. In Figure 1, two continuums are drawn. The first is a continuum moving from abstract to concrete. "Abstraction" has come to have multiple interpretations (Brachman, 1983). For our purposes, we consider *abstraction* as presented in Locke's Essay Concerning Human Reasoning.

> Words become general by being made the signs of general ideas; and ideas become general by separating from them the circumstances of time and place, and any other ideas that may determine them to this or that particular existence. By this way of abstraction they are made capable of representing more individuals than one; each of which having in it a conformity to that abstract idea, is (as we call it) of that sort (type).

The other continuum is from deep to surface structure. Perhaps the best explanation is by example. Chomsky (1971) formalized the notion that the surface grammar of any given instance of a language is a manifestation of a *deep structure.* The deep structure, in this case, is a basic grammatical system from which the whole variety of manifest *surface structures* (actual communication in a language) can be generated. The axiomatic level, to be discussed shortly, is a deep structure.

A system has *system structure* and *system function.* Structures themselves do not function, but systems function because they have the structure to do so. One can infer, only in part, one from the other. An effective axiomatic model should abstract both notions of system structure and system function. The distinction between system

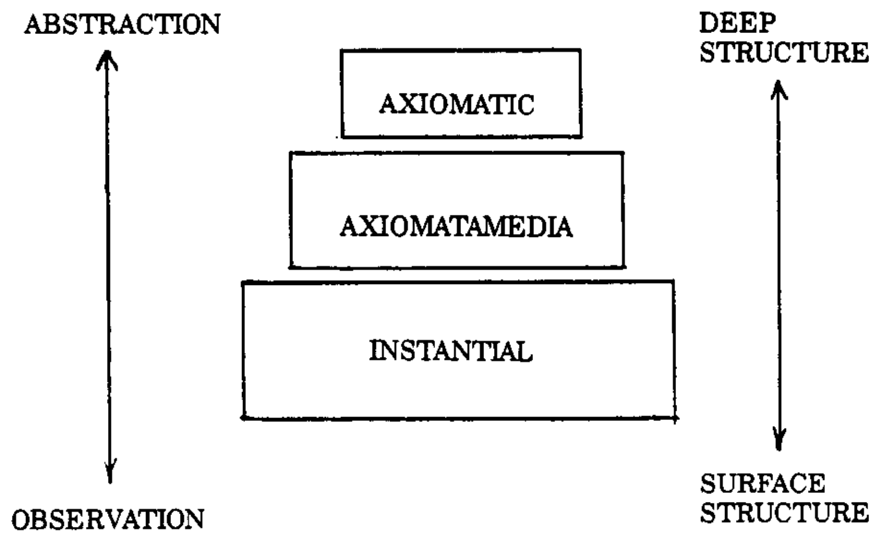


Figure 1

A Conceptual Schema for Metasystems

structure and system function is analogous to the distinction between language syntax and semantics.

A *target system* is the system being described at the instantial level, and so is a description of an existing or proposed implementation of an information system (IS).

An *axiomatic model* or system has acquired two basic interpretations, the more traditional being that found in mathematics where a set of axioms are used to formally prove the truth of an assertion. The second interpretation of an axiomatic model comes from the empirical sciences; the truth of an axiomatic model is judged, over time, on the basis of its power to explain observable phenomena. Pascal referred to this as deductive synthesis—rather than premises extending to consequences via formal proof, the truth of the premises "rebounds back" from the consequences. This second interpretation of an axiomatic model is the one used in this discussion.

With axiomatic models derived by deductive synthesis, completeness of the axioms is shown pragmatically over time. That is, the completeness of the axioms is initially governed by past experience and insights. Incompletenesses are later discovered through use of the axiomatic model in practical application. The axiomatic models presented here, therefore, are hypotheses.

In light of the above definitions, the *axiomatic level* (model) is a deep structure arrived at by abstraction that is used to define a description language at the median level which, in turn, is used to describe specific target systems at the instantial level. *All* three levels, taken together, form the system description model.

A *metasystem*, for purposes of the present discussion, is an computerized implementation of an axiomatic model that provides facilities for the definition and analysis of median and instantial models. The current discussion concentrates on the description of complete and consistent system descriptions and is not concerned with metasystems used to generate executable programs (Cameron and Ito, 1984).

A schematic of a typical metasystem is shown in Figure 2. Given the concepts and facilities provided at the axiomatic level, a system description language is defined giving the median level view. Target systems are defined using the descriptive forms defined at the median level and so form the instantial model.

Levels in a Metasystem: A General Description

Figure 1 depicts the conceptual structure of a typical metasystem. Aspects of metasystems that the figure attempts to illustrate are:

1. The levels in a metasystem build from each other. The model at a lower (more surface) level inherits concepts from the levels above

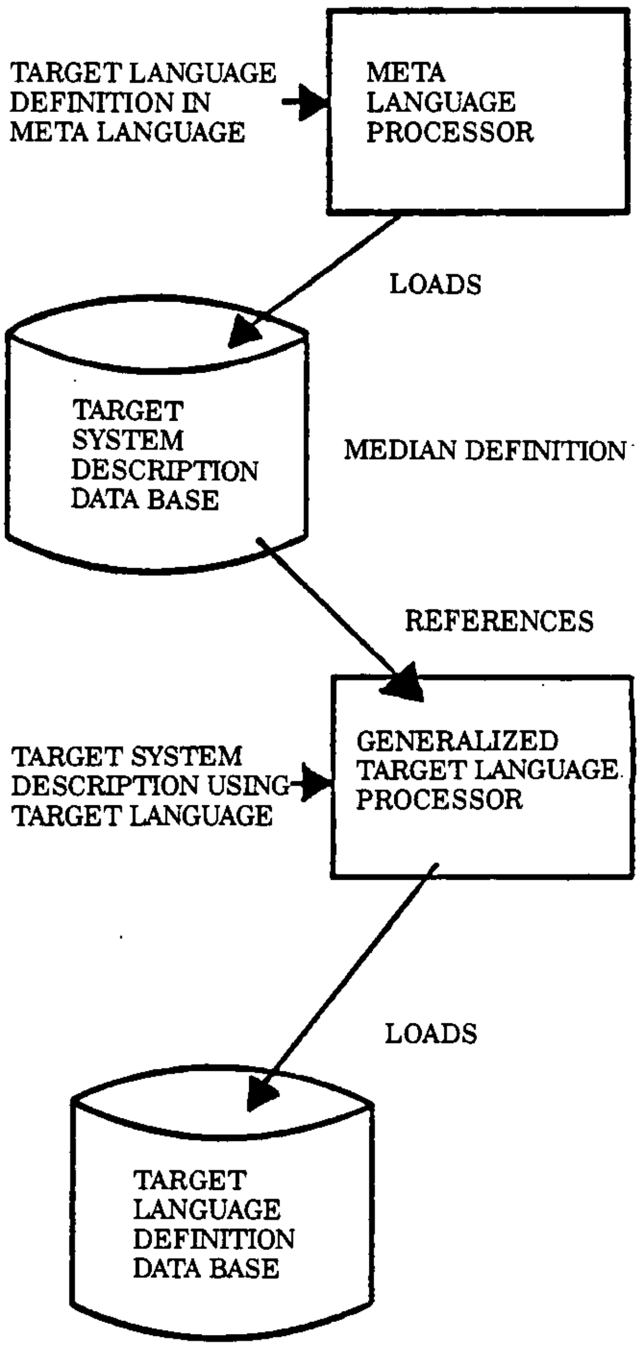

TARGET LANGUAGE DEFINITION IN META LANGUAGE
META LANGUAGE PROCESSOR
LOADS
TARGET SYSTEM DESCRIPTION DATA BASE
MEDIAN DEFINITION
REFERENCES
TARGET SYSTEM DESCRIPTION USING TARGET LANGUAGE
GENERALIZED TARGET LANGUAGE PROCESSOR
LOADS
TARGET LANGUAGE DEFINITION DATA BASE

Figure 2

A Schematic of a Typical Metasystem Implementation and are a refinement of the model at the upper level(s) (e.g., the axiomatic term "entity" is used to type the median term "file" which, in turn, is used to type the instantial term "master customer file."). As shown in a later section, the Plexsys metasystem (Konsynski and Nunamaker, 1982; Kottemann, 1984; Stott, 1984) involves the inheritance of context dependent system roles from the axiomatic to the median, and the median to the instantial levels.

2. Although lower levels inherit from upper levels, they may choose to disregard concepts of upper levels (i.e., the levels may be disjoint). (E.g., the instance level modeler may choose to override a completeness rule specified at the median or axiomatic level.)

3. The lower the level the less abstract the model and the more it is descriptive of observable phenomena. (E.g., "master customer file" is a more detailed description of the target system than is "file" or "entity.")

4. Each level may indeed have levels of abstraction within it. (E.g., within the median level itself, "file" may be further refined into "master file" and "transaction file.")

5. Lower level models contain more information, in the information theoretic sense, than upper level models (lower level models are more verbose). The single term "entity" is refined into a multitude of median terms such as "file," "data item," "process," and "output report." File, then, is refined into a multitude of instances of files—"customer file," "product file," and "employee file."

6. The completeness and consistency at upper levels determines the *intrinsic* completeness and consistency of models at lower levels. *This point is the focus of the present discussion.*

Given the previous list, particularly number 4, it is apparent that the breakdown of a metasystem into three levels is not so much due to a law of nature, but rather is due to the fact that there are differing implications, motivations, and goals at each level. Also, the model definers at each level are concerned with different aspects of the world being modeled: the axiomatic level—to define fundamentals of target systems in a set of development settings; the median level—to define the model tailored to specific development settings; the instantial level—to define the target system itself. Finally, the three tier view is useful for purposes of discussion and for the realization of a metasystem implementation.

For purposes of discussion, models at all levels are stated in a language. These languages are composed of *terms* and *expressions*. As noted, the higher levels are abstractions and thus a term at one level becomes a definition of a term type at the next lower level. The relations among, and motivations at the respective levels are as follows.

As shown in Figure 2, the concepts at the axiomatic level, manifest in the meta language provided by the meta system, are used to define a target system description language at the median level. That median language is then used to describe actual target systems at the instantial level.

The axiomatic level provides abstract terms (axiomatic terms) which become term types at the median level. Example axiomatic terms are "entity" and "relation." All language terms defined at the median level then, are declared as either type "entity" or type "relation." *Language statement constructs*—median expressions—to be used at the instantial level are also defined: e.g., "Process instance-slot-1 produces file instance-slot-2."

Using axiomatic concepts, and corresponding metasystem facilities, the language definition may also include rules of composition to be imposed at the instantial level. For example, a rule might be defined to assure that a given instance of type "file" is produced by one and only one "process" instance. Such rules are specified with the goal of insuring the integrity of instantial models—the descriptive models of target systems. As noted, the thrust of the present discussion is the degree to which the axiomatization can insure integrity of the median and instantial levels.

In addition, some axiomatic terms and concepts may be understood at the axiomatic level, for example, "is part of," and "is a subtype of." Thus, the meaning of "A is a subtype of B" is understood at the axiomatic level. In the metasystem, this understanding means that the metasystem has predefined operations invoked to act appropriately in the case where A is a subtype of B. One operation may be the inheritance of properties of A by B.

Finally, at the instantial level, the language defined at the median level is used to describe the target system (e.g., "Process Employee-update Produces File New-employee-file.")

As discussed in the following sections, one criteria for evaluating or constructing metasystems is the degree to which its axiomatic level can understand the meaning of abstract definitional terms and enforce resultant generalized rules of system integrity. This implies that there exists an ideal axiomatic model: one that is abstract enough to cover a large set of development settings yet concrete enough to make meaningful, operational distinctions of "what an information system is" at the axiomatic level.

Requirement for a Metasystem

Although each level in a metasystem can overcome definitional deficiencies in upper levels, it is argued here that the power of a metasystem is quite dependent on the power of the axiomatic level. Indeed we define the ideal axiomatic model as a deep structure arrived at by abstraction that is used to generate models at the median and instantial level *and* that ensures intra-and inter-level completeness and consistency. Such completeness and consistency concerns are evident in any requirements or design specification, for example, database specification (Brodie, 1983).

Four characteristics that influence the power of a metasystem are:

1. Scope of the axiomatic model—Scope is defined as the variety of median levels that the axiomatic model is capable of generating. In short, scope is the variety of the development settings addressed.

2. Granularity at the axiomatic level—Granularity is a measure of the fundamental distinctions intrinsic at the axiomatic level. An operational definition is the number of axiomatic terms and concepts. These, then, are axiomatic concepts that are understood and operationalized by the metasystem.

3. Succinctness (efficiency) of language definition—Succinctness concerns the ease with whcih the median model can be defined. Note that in the median definition the language to be used at the instantial level as well as rules to be imposed in the formulation of instantial models may be defined. Succinctness can be defined via information theory and the related metrics of software science (Halstead, 1977).

4. Dynamism of the median and instantial levels—Target systems *and* views of system development change over time. It is desireable for a metasystem to allow changes to the median model and that those changes propagate to the instantial models generated from it. This characteristic is largely a metasystem implementation issue and is discussed in a later section.

There are several implications of, and relations between the concerns just listed. In short, a conflict stems from the fact that too drastic an abstraction looses much of the general semantics of systems. Indeed, total abstraction would leave us with one single axiomatic term—"everything." A certain degree of abstraction at the axiomatic level is obviously desirable. As we abstract we gain scope. As we abstract, however, we also sacrifice granularity and succinctness. Referencing the preceding list, the major advantage of abstraction is to broaden the scope of the metasystem. If, however, the axiomatic level is too abstract:

1. The less the axiomatic level can "understand" the meaning of terms and expressions at the median and instantial levels, consequently

2. The less powerful can be axiomatic analysis—the metasystem provided integrity analysis of the median and instantial models—and

3. The more distinctions must be made at the median, language definition level. This implies that numerous completeness and consistency rules must be specified at the median level. In short, succinctness is sacrificed. Further, the definer of the median level is left wondering if all requisite rules have been defined.

It is desirable for the metasystem to understand various aspects of general system structuring and function, or behavior. As proposed in general systems theory, see for example (Checkland, 1981), certain features of systems are omnipresent. Some areas in which the generalizations exist include:

1. Distinct roles played by terms in a system,

2. Role completeness rules—all necessary roles filled,

3. Role consistency rules—functional dependencies among processes, for example,

4. Temporal aspects of system behavior, and

5. Purpose (goals) of processes.

One goal of a metasystem is to axiomatize not only the "syntax" of systems, but also the "semantics" of system functioning. In the following section, three metasystems are selected for discussion.

Existing Metasystems: SEM, SDLA, and SDS

Existing metasystems provide quite abstract axioms. Specifically, the components of a system are abstracted into two or three axiomatic terms. All are closely allied to the Entity-Relationship-Attribute model proposed for logical database design (Chen, 1976).

SYSTEM ENCYCLOPEDIA MANAGER (SEM)

SEM (Teichroew, Macasovic, Hershey, Yamamoto, 1980) was developed under the auspices of the ISDOS project at the University of Michigan. It is used primarily to implement the PSL (Teichroew, Hershey, 1977; Teichroew and Gackowski, 1977) system specification language, but has also been used to implement languages for office (Konsynski and Bracker, L., 1982) and network (Konsynski and Bracker, W., 1980) specification. An architecture similar to SEM has been proposed for an Information Resource Dictionary System (Kerschberg, *et al.*, 1983). SEM uses as its axiomatic descriptive terms "object," "relation," and "property." This general model was proposed as a logical database design modeling form by Chen (1976)—the Entity-Relationship-Attribute model.

The first version of SEM (Yamamoto, 1981) allowed for the definition of:

1. Terms to be typed as objects (entities), relations, or properties (attributes).

2. Statement forms which are strings of terms.

3. The structure type (n-ary cardinality) associated with the relation in a statement form. These cardinality structures involve up to 4-ary relations.

The median model is defined using the Information System Language Definition System. The following is a sample definition of a language subset for describing information systems. Details, such as the definition of synonyms for objects, are omitted for clarity.

Objects are defined by,

```
OBJECT agent;
OBJECT process;
OBJECT report;
```

Properties are defined by,

```
PROPERTY average-size;
   APPLIES report;
   VALUES INTEGER 0 THRU 1000;
```

Relations and their respective n-ary cardinalities are defined by,

```
RELATION performs;
   PARTS agent-part, process-part;
   COMBINATION agent-part, agent
      WITH process-part, process;
   CONNECTION-TYPE S4;
   CONNECTIVITY ONE agent, process;

RELATION produces;
   PARTS process-part, report-part;
   COMBINATION process-part process
      WITH report-part report;
   CONNECTION-TYPE S2;
   CONNECTIVITY ONE process-part
                     MANY report-part;
```

The actual statement forms to be used at the instantial level are given by,

```
STATEMENT performs-statement
   USED agent-part performs;
      FORM performs process-part;
   USED process-part performs;
      FORM is performed by agent-part;

STATEMENT produces-statement
   USED process-part produces;
      FORM produces report-part (report-part);
   USED report-part produces;
      FORM is produced by process-part;
```

Finally, example terms and statements defined at the instantial level are given by,

```
DEFINE AGENT chicago-processor;
DEFINE PROCESS sales-reporting;
DEFINE REPORT salesman-performance;
DEFINE REPORT regional-sales-performance;

AGENT chicago-processor
   PERFORMS PROCESS sales-reporting;

PROCESS sales-reporting
   PRODUCES REPORT salesman-performance,
                   regional-sales-performance;
```

The axiomatic level in SEM is limited. Instead, specification of integrity rules are relegated almost entirely to the median level. The robustness of rule types, moreover, is limited largely to the specification of cardinalities for n-ary relations and static type checking.

In the previous example, the instance chicago-processor can only be used in relationships defined for type AGENT. Static typing is typing that is context independent. That is, despite the use of "chicago-processor" in the instantial model, it is consistently typed as AGENT.

An extension to SEM was later proposed to support the specification of integrity rules at the median level (Kang, 1982). These rules are specified in predicate calculus.

1. Implication-derivation—if an object exists in a given part of one relation then it can be assumed that it exists in a given part of another relation. Thus if x is a parent of y, and y is a parent of z, then x must be a grandparent of z.

2. Mutual exclusion—an instance of an object may exist in only one of a set of possible relations. Thus, a person cannot be both married and single.

3. Exclusion—if an instance is in one relation it may next participate in a limited set of other relations. Such rules constrain state transitions of a description. For example, marrital status, once given the value of "married," may be next assigned the values of "divorced" or "widowed" but not "single."

The later specification of SEM, although allowing for a substantial degree of integrity definition, has not abstracted many more concepts into the axiomatic level. The definition of the above integrity rules must be given at the median level. Further, SEM stores median and instantial models in separate databases. *Any* median level modification necessitates regeneration of the databases. Also, if a median term is deleted, the change does not propagate to the instantial model. In summary, SEM scores well on scope and poorly on dynamism, granularity of the axiomatic level, and succinctness of median level definition.

SDLA

SDLA (Knuth, *et al.*, 1979; Demetrovics, *et al.*, 1982) was developed, in part, at the Hungarian Academy of Sciences. Our discussion of SDLA is brief as only differences between SEM and SDLA are highlighted. The major differences between SEM and SDLA include:

1. The distinction between "relation" and "entity" is abstracted into "concept." A concept is associated with attributes.

2. SDLA allows abstraction at the median (language definition) level. If a concept2 is a subtype of a concept1, concept2 inherits the attributes associated with concept1.

3. Axioms of system structure are refined.

Depending upon one's view of a term, a term may be treated as an object or a relation. The concept of "communication," for example, may be used as a relation —*John is in communication with Judy*—or as an object —*communication is difficult.* In SDLA, this problem is alleviated, not by allowing multiple roles, but by abstracting away the distinction between objects and relations to form "concepts."

As noted in a previous section, the conceptual levels in a metasystem may indeed contain levels. Unlike SEM, SDLA axiomatizes abstraction to be used at the median, language definition level. Thus, in an information system there may be the concept "file." There exist, moreover, types of files such as disk files and print files. In SDLA, the definer of the median level may define a type "file" with the attribute "size" and the subtypes "print file" with the attribute "average number of pages," and "disk file" with the attribute "average number of bytes." The types "print file" and "disk file" inherit the attributes of their supertype and thus also have the attribute "size." Language statement forms may be defined that include super or subtypes. Thus the statement form "FILE x IS-LOCATED-AT y" allows instances of type "file," "print file," or "disk file" to participate. The statement form "PRINT FILE x IS-LOCATED-AT y," on the other hand, constrains the instances to be of type "print file."

Lastly, SDLA allows for the specification of structure types for binary relations. These integrity rules are similar in effect to cardinality rules; they constrain described system structures. The concept "parent-child (human, human)," for example, is undeniably antisymmetric (i.e., my mother cannot also be my daughter). SDLA allow for dinary relational property specification to enforce structures to be:

antisymmetric,
irreflexive,
hierarchic,
precedence, and
lattice,

where "hierarchic" implies a tree structure, "precedence" implies a structure with a transitive closure that is a partial ordering, and "antisymmetric," "irreflexive," and "lattice" are used in their algebraic sense.

SDS

SDS (Levene and Mullery, 1982), as with SEM, uses an entity/relationship scheme. In SDS terminology, entities are termed "components," which may have attributes termed "properties," and components are associated via "relationships."

The SDS system is weak with respect to the axiomatization of and availability of facilities for median definition of integrity rules. The primary support for integrity analysis is a query facility. That is, a user may query the requirements database in an attempt to discover integrity violations.

Conclusions on Some Current Metasystems

The development of metasystems for system development lends a degree of flexibility to the application of formal system specification tools. A metasystem represents an axiomatic view of information systems. The contribution of a metasystem rests not only on the computerization of tools for system description but also in the power of the meta view of information systems, particularly: what are the essential abstract elements in an information system that dictate its integrity? In this sense a meta paradigm represents a theory of information systems.

The meta approaches described above go far toward realizing both abstract formalism of information systems and the facilities for automation of IS specification languages. The approaches, however, abstract at the axiomatic level to such an extent that integrity rules cannot be axiomatized. Given the SEM entity-relation axioms, for example, virtually all integrity rules must be explicitly defined at the median level. Axiomatized integrity checking at the median level is limited to rules such as "a median term typed as a relation cannot also be typed as an entity," and "all median terms should be used in at least one statement form." Axiomatized integrity checking of the instantial level is limited analogously. However, at the instantial level the statement forms defined at the median level imply integrity constraints. In the example given for SEM, if an instance term is defined as median type "process," that instance term may only be specified in relations (and associated statements) containing "process."

In the following section, an alternative meta paradigm is developed with the intent to provide an aziomatic model with sufficient granularity for generalizing integrity rules and integrity verification for *both* the median and instantial models generated using the metasystem.

The Plexsys Meta Paradigm

The Plexsys project is an ongoing effort to realize a development environment that provides methodological and computer-aids for IS development (Konsynski and Nunamaker, 1982; Kottemann, 1984; Stott, 1984). The Plexsys computer-aids are both active and passive, where active tools actually perform system design activities. The emphasis here is on the relatively passive tools for system description.

AXIOMATIC TERMS AND CONCEPTS

As asserted in the previous sections, the distinctions drawn at the axiomatic level of a metasystem should be abstract enough to address a large number of median and instantial models while also granular enough to formalize and axiomatize basic aspects of system structure and functioning. As a side benefit of granularity, the more distinctions *intrinsic* to the axiomatic level, the fewer distinctions, or integrity rules need be explicitly stated at the median level, hence the more succinct the media (language) specification. Indeed, as shown below, generalizing one type of integrity rule may circumvent the need to explicitly state a large number of manifestations of the general rule type at the median level. A second major benefit of granularity at the axiomatic level is that it potentially allows for integrity checking of the median model, that is, the metasystem can perform integrity checking of the language definition.

Many authors have proposed axiomatizations of systems, these axiomatizations being a list of aspects shared by all systems. Checkland (1981), for example, gives the systems elements as

Transformation process
Ownership of the system
Actors in the system
Customers of the system
Environmental constraints

Nadler (1975, 1981), gives:

1. Function—The mission, purpose, or primary concern of the system.

2. Inputs—The physical, information, or even human items that the system must recognize and handle.

3. Outputs—The physical, information, or human items, both desirable and undesirable, which the system will predictably produce from the given inputs.

4. Sequence—The step-by-step process by which acceptable inputs are transformed into predictable outputs.

5. Environment—The physical, and psycho-socio-logical setting in which the system operates.

6. Physical—The physical resources that are catalysts used in each step of the sequence but are not part of the output.

7. Human—The human resources that are agents used in each step of the sequence but are not part of the output.

8. Information—The information which is used in each step of the sequence but is not part of the output.

Although the axiomatization choosen for the Plexsys meta view is influenced by taxonomies such as those above, it draws most heavily from work in linguistics: specifically, in deep case analysis (Bruce, 1975). It differs from the meta approaches discussed above in that 1) it is axiomatically more granular, and 2) it treats the role of median and instantial terms context-dependently. Context-independent, or static typing involves the invariant correspondence between instantial, median, and axiomatic terms. Thus, a "file" is statically typed as "entity," and "employee file" is statically typed as "file." Context-dependent typing involves the variable *roles* of terms. For example, a "file" may be produced as a result of a process and may be used as an input to another process.

CASES IN LINGUISTICS AND ARTIFICIAL INTELLIGENCE

In a natural language such as English, words assume different cases or roles in different sentences. In "the man sees the dog," the dog is the object of the sentence. In "the dog sees the man," the man is the object. Whereas in languages such as Latin the role of a word in a sentence is manifest as a surface case—different spelling of the word for different roles. Languages such as English rely on word order, prepositions, and other similar implication mechanisms.

Despite the manifestation, or lack thereof, of surface cases that indicate the role of works in sentences, linguists such as Fillmore (1968, 1971) note the presence of deep case structures. Take, for example,

Bob wrote a letter with a pencil.

Bob is the ACTOR of the verb wrote, the letter is the OBJECT, and pencil is the INSTRUMENT. Note that although the sentence

The note was written with a pencil by Bob.

has a different surface structure, the cases, or *roles*, of Bob, note, and pencil remain the same. In the sentence

Judy poked Bob with a pencil.

although Bob may be statically typed as a "human," the role assigned to Bob is now OBJECT. Thus, while type is context-independent, role is context-dependent. It is this notion that allows for axiomatization of integrity rules.

A case formalism represents an attempt to abstract out the essences of language semantics. The challenge in delineating a case formalism is similar to that in forming the axiomatic level—to maximize the captured semantics with a minimum of distinctions or cases. The number of proposed roles, or cases vary between five and thirty (Bruce, 1975). They are used both for linguistic analysis and natural language processing (see Schank (1975) for an example of the use of case grammars in natural language understanding systems). Here, the purpose in developing a case formalism is to derive an axiomatic model that affords a generalization of integrity rules for models at both the median and instantial levels.

USE OF CASES IN PLEXSYS

A case is a role type that a given term assumes in a given context or language statement form. Although a given median term (e.g., "Bob is a human,"), it assumes different roles in different contexts (e.g., Bob is an ACTOR in one context and and OBJECT in another). The role types chosen for the axiomatic level in Plexsys are:

1. *Process:* an action that changes the state of the system under investigation, (e.g., performing a *sales assessment task*).

2. *Actor:* the performer of a process (e.g., *sales manager* performs sales assessment).

3. *Instrument:* the term is used by an actor to perform a process. A term in the role of an instrument is *not* changed by the process (e.g., sales manager performs sales assessment using the *sales performance report*).

4. *Directive:* the term(s) that control(s) the process—these may be constraints and/or objectives (e.g., sales manager performs sales assessment to *increase sales performance*).

5. *Material:* the term that is changed by a process in its state before the change (e.g., a *file to be updated*).

6. *Result:* the term that comes into existence as a result of the process. In a different context, the term that is a result here may be a process, actor, instrument, directive, or material (e.g., *an updated file).*

7. *Temporal qualification* roles: These include—*momentary verbs* that state transition (e.g.,

"stop processing"), and *continuative verbs* that represent states (e.g., "is processing").

8. *Spatial qualification* roles include: *locus*: where a term resides, and *movement*: which infers the existence of source and destination loci, speed, and process to move the term, which in turn, infers the existence of instrument (transport medium), actor, directive, material (locus before move), and result (locus after move).

9. *Object*: a term whose existence or structure is merely being considered—not used in a process behavior description context (e.g., "The sales coordinator is human," "Disk file is a type of file," and "File has a size.)"

In defining the median level—the language to be used at the instantial level for instance modeling—a language construct is defined. A language contruct is a series of axiomatic terms, median terms, and *slots* for instances: Manager instance-slot has-authority-over manager instance slot for-task instance-slot-3.

For each median term that is to be associated with an instance slot, the allowable string format of the instance term is defined. Thus, example instance name formats for median terms are defines as:

Aim: string of printable characters
Task: string
Agent: string
Information set: string
Priority: enumerated string (1,2,3,4,5,6,7,8,9)
Frequency: enumerated string (daily, weekly, monthly).

For each language construct two types of specifications are defined: context-independent and context-dependent typing. One, the median type to be checked for, or assigned to an instance defined in that slot is defined. In the above, slot-1 would be associated (typed) as "Manager." Two, the role of the median terms, and the role assumed by the instance when it is placed in a corresponding slot, is defined.

Task: Process
  Realizes Aim: Referent
  With Priority: Object

Agent: Actor
  Performs Task: Process
  With Frequency: Momentive verb

Task: Process
  Is Performed At Location: Spatial Qualifier

Agent: Process
  Uses Information Set: Instrument
  To Perform Task: Process

At the instantial level, then, statements such as the following may be defined:

Task Check-Delinquents
  Realizes Aim Reduce-Bad-Debt
  With Priority 5.
Agent Collections-Manager
  Performs Task Check-Delinquents
  With Frequency Monthly.
Task Check-Delinquents
  Is Performed At Location Tucson-Office.
Agent Collections-Manager
  Uses Information Set Aging-Report
  To Perform Task Check-Delinquents.

AXIOMATIZED INTEGRITY RULES

Granularity of the axiomatic level and the use of context-dependent roles provides the potential for axiomatizing various types of integrity rules. There are two varieties of axiomatic integrity rules—those whose violation indicate a definite integrity violation. Given the role formalism proposed in the preceding section, the following integrity rules for the language definition can be axiomatized.

1. Except for the special case of sources and sinks, all median terms that exist in a process role must be associated with terms in the roles of actor, directive, material and/or instrument, and result. This rule is assuring complete process description, that is, a process must be performed by some actor for some purpose, using some inputs and/or instruments to produce some outputs.

2. A median term in the role of result should also be in the role or process, actor, instrument, material, or directive. This rule asserts that a result must serve a useful purpose in another context.

3. A median term in the role of process, actor, instrument, material, or directive should also be in the role of a result. This rule is the inverse of rule 2 and asserts that something of use must come from somewhere.

4. All median terms should be in the role of result in only one statement construct. This rule asserts that nothing is produced by two disparate median types. Tangentially, it also helps assure unambiguous typing.

5. Existence of a median term in the role of material or instrument should be associated with a process that uses it. This rule asserts that a material or instrument is useful only as used by a purposeful process.

6. Existence of a median term in the role of result should be associated with a process that produces it.

7. All terms in the role of process should have temproal qualification. This rule asserts that all processes take place in time, and are interrelated.

8. The language definition should have a language construct to associate processes to loci. Movement of results is inferred by the locations of processes.

The preceding integrity rules can be interpreted as utility and possibility rules—that a system never does anything if there is no use for the action, and that a system must do something to produce necessary elements in the system. Using the role paradigm, the above integrity rules can be analyzed for an arbitrary language definition. As outlined in an above section, a metasystem should have the capacity for a lower level to disregard concepts at an upper level. This is accomplished via the option of naming certain roles as "implicit." Thus, for example, the directive for a process type "Decision Making Task" may be associated with "Directive is Implicit." In this case the directive for a decision making task need not be specified at the median or instantial levels.

The rules for the median level have analogs at the instantial level. These are as follows:

1. Except for the special cases of sources and sinks, all process instances should have associated instantial terms in the roles of actor, instrument and/or material, result, and directive. This insures a complete process description.

2. An instantial term in the role of a result should also be in the role(s) of process, actor, instrument, material, or directive in a different context. This insures utility of process outputs.

3. An instantial term in the role of process, actor, instrument, material, or directive should be in the role of a result in another context. This insures that nothing comes from nowhere.

4. Existence of an instantial term in the role of material/instrument or result implies the existence of a process that uses or produces respectively.

5. All instantial terms should be in the role of a result in one and only one instantial statement—two distinct terms do not produce the same thing.

6. All instance terms in the role of result must be associated with a process that produces it—a result must be produced by some process.

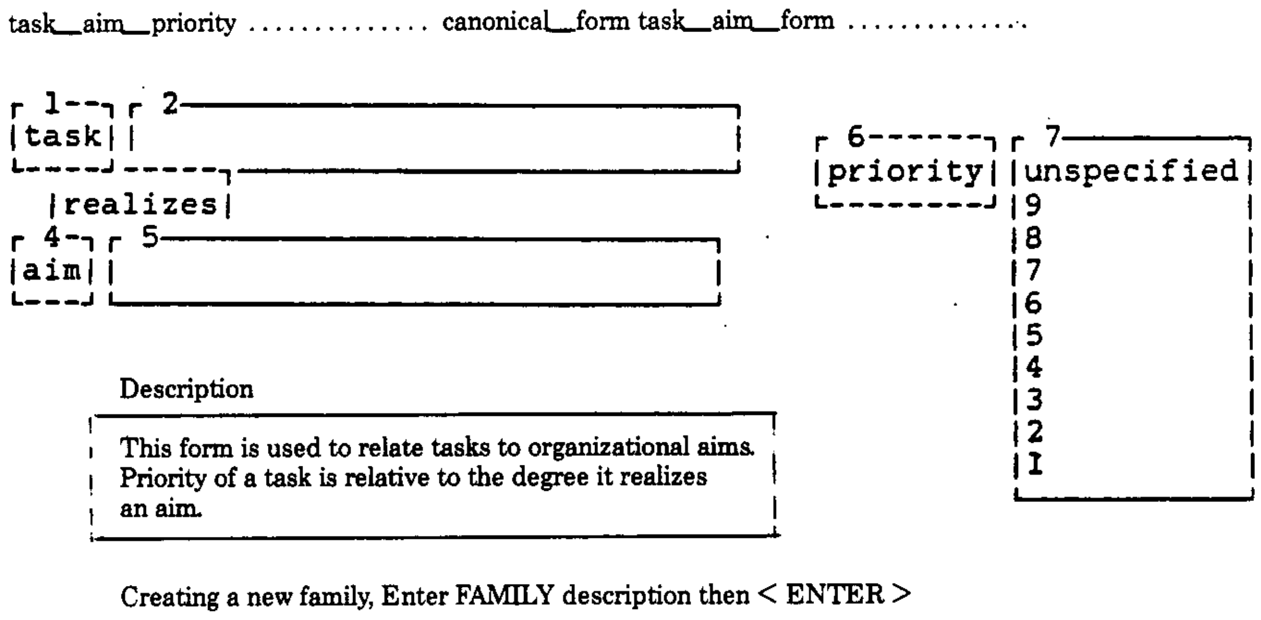
7. All instance terms in the role of process must have temporal qualification—a process takes place in time.

8. All processes should have a location. Further, if A produces X and B uses X, the existence of a process to transport X from the locus of A to the locus of B is inferred.

Note that if the language definition satisfies the median level integrity rules given above, integrity checking of the instantial model is straight-forward. For example, the first median rule insures that there are statement forms for associating a given median type that is extant in the role of process with actors, directives, results, etc. Roles are inherited from median terms to their corresponding instance terms. At the instantial level, then, the integrity checking reduces to the verification that an instance of a given median type is extant in all statement forms containing the median type.

Completeness and consistency are closely related to the notion of ambiguity in language specification and use. The rules above *do not* insure total integrity. As for ambiguity, there are differing concerns between specification of an algorimthmic language, such as Pascal, and systemic languages. With specification of algorithmic languages, it is critical that no ambiguities exist, thus necessitating the use of a context sensitive grammer. With respect to IS modeling, however, a certain degree of ambiguity is indeed necessary due to the continual reality of limited system knowledge. The Plexsys axiomatization presented above only partly insures unambiguity in the median and instantial models.

All of the rules just given are integrity rules that are generalized into the axiomatic level of the Plexsys system. Descriptions of target systems at the instantial level expressed in an arbitrary language can be analyzed by the system for violations of the rules given above *without* the need to identify and define the rules. The integrity analyzers and other Plexsys tools are discussed in the following section.



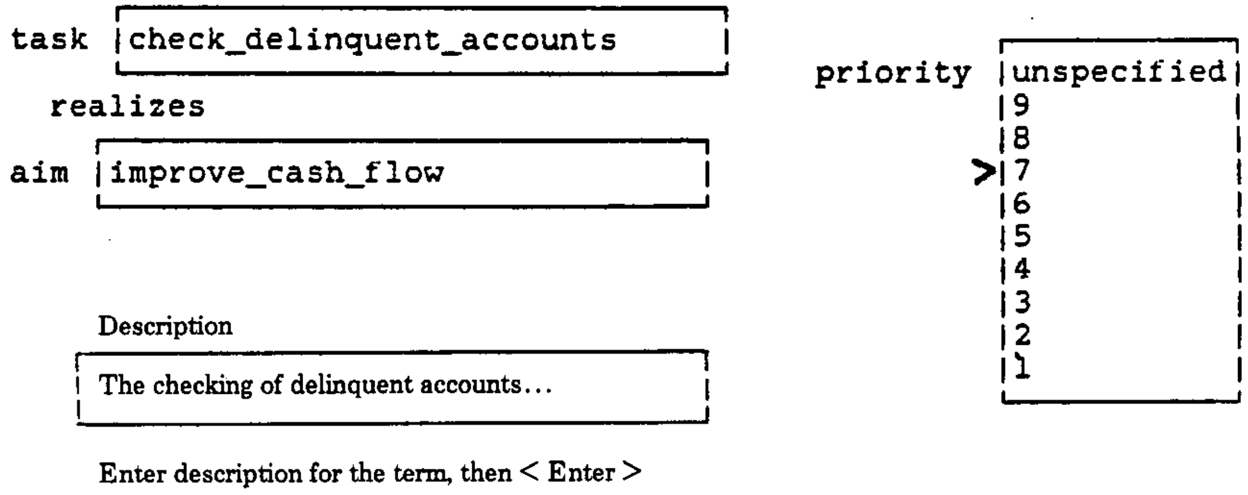
```
task_aim_priority .............  canonical_form task_aim_form .............

┌ 1--┐ ┌ 2──────────────────────────────────┐
|task| |                                    |
└----┘ -----┐──────────────────────────────┘        ┌ 6------┐ ┌ 7───────────┐
   |realizes|                                       |priority| |unspecified|
┌ 4-┐ ┌ 5──────────────────────────────────┐        └--------┘ |9          |
|aim| |                                    |                   |8          |
└---┘ └────────────────────────────────────┘                   |7          |
                                                               |6          |
                                                               |5          |
      Description                                              |4          |
    ┌──────────────────────────────────────┐                   |3          |
    | This form is used to relate tasks to organizational aims.|2          |
    | Priority of a task is relative to the degree it realizes |1          |
    | an aim.                              |                   └───────────┘
    └──────────────────────────────────────┘

      Creating a new family, Enter FAMILY description then < ENTER >
```

Figure 3

A Statement Form Definition in Plexsys

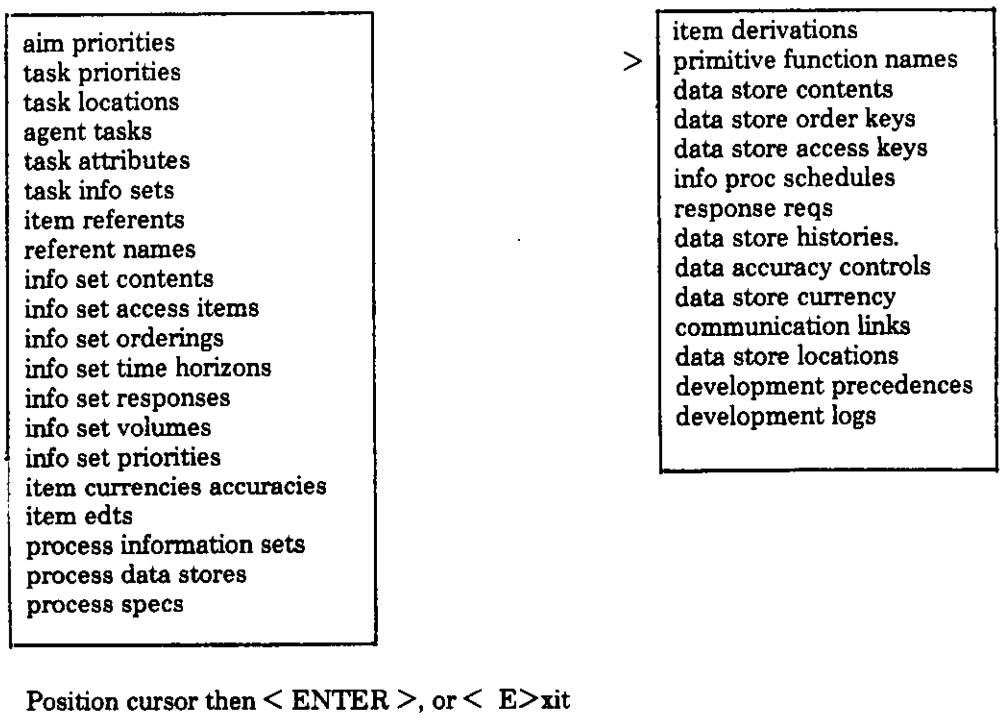


```
        ┌──────────────────────────────────┐
task    |check_delinquent_accounts         |
        └──────────────────────────────────┘      ┌───────────┐
  realizes                                priority |unspecified|
       ┌──────────────────────────────────┐        |9          |
aim    |improve_cash_flow                 |        |8          |
       └──────────────────────────────────┘       >|7          |
                                                   |6          |
                                                   |5          |
                                                   |4          |
    Description                                    |3          |
    ┌─────────────────────────────────────┐        |2          |
    | The checking of delinquent accounts...|      |1          |
    └─────────────────────────────────────┘        └───────────┘

    Enter description for the term, then < Enter >
```

Figure 4

Defining an Instance Statement in Plexsys


aim priorities
task priorities
task locations
agent tasks
task attributes
task info sets
item referents
referent names
info set contents
info set access items
info set orderings
info set time horizons
info set responses
info set volumes
info set priorities
item currencies accuracies
item edts
process information sets
process data stores
process specs

item derivations
> primitive function names
data store contents
data store order keys
data store access keys
info proc schedules
response reqs
data store histories.
data accuracy controls
data store currency
communication links
data store locations
development precedences
development logs

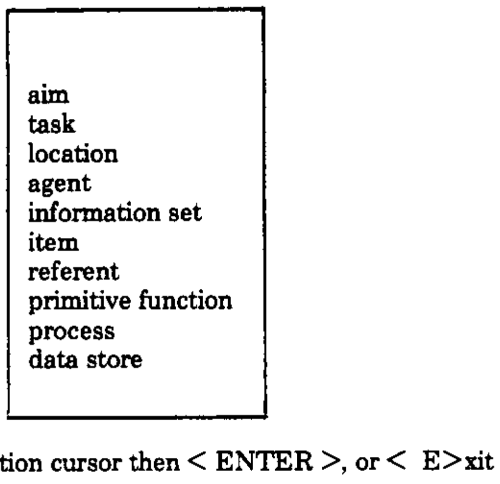
Position cursor then < ENTER >, or < E>xit

Figure 5

Menu of Statement Forms in Plexsys

The Plexsys System Implementation

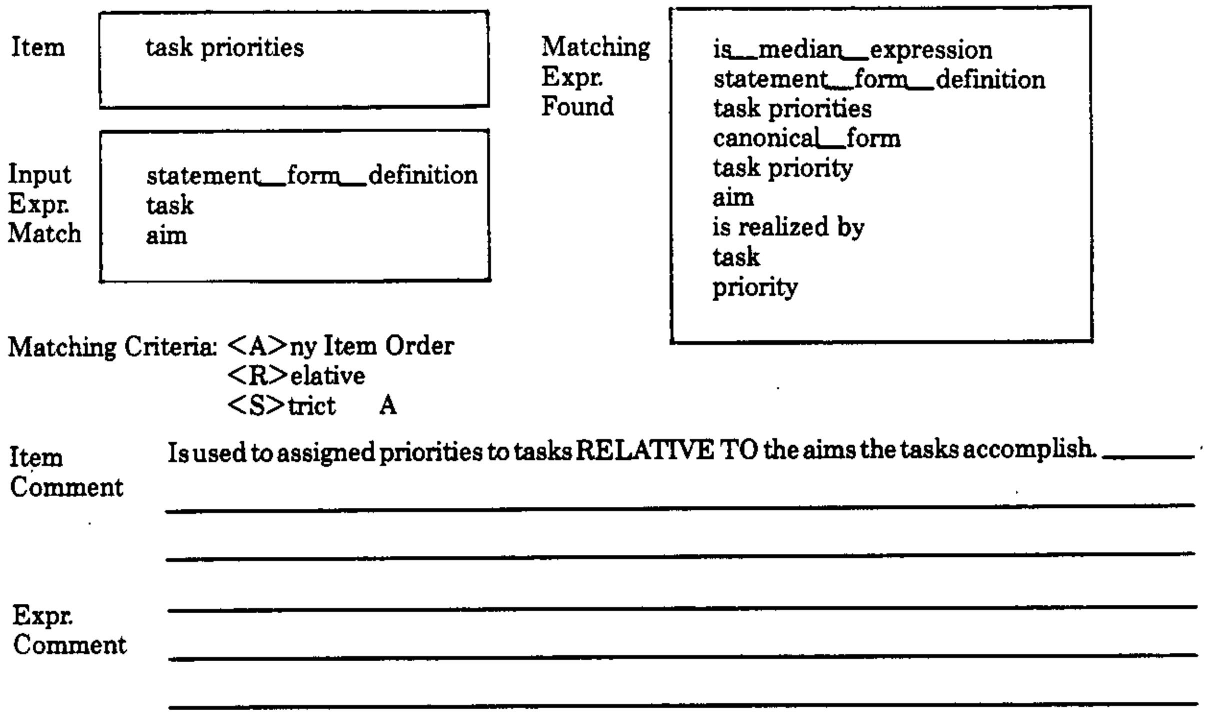
In Plexsys, a special purpose database system, JAMES, has been designed and implemented for storing and manipulating symbolic knowledge and text. It is similar in some ways to the internal storage structure used in a typical implementation of a LISP interpreter (Siklossy, 1976). One JAMES database is used to store all model levels including axiomatic terms and expressions, the definitions of median terms, and descriptive language definitions as well as the actual target system descriptions. All expressions are prefixed with terms that indicate the level and type of the expression. For example, statement form definitions are prefixed with the terms "is-median-expression" and "statement-form-definitions." Given that the description language definition and the target system description language are stored in the same database, modifications to the target system description as well as modification to the language definition itself may be made throughout the development process. This differs from current approaches in that 1) the current database does not need regeneration upon language modification, and 2) language definition modifications automatically propagate to the target system description. By providing a dynamic language environment, Plexsys allows not only customization to a given problem domain but also allows for customization over time—as more is learned of the language required during the development process.


aim
task
location
agent
information set
item
referent
primitive function
process
data store

Position cursor then < ENTER >, or < E>xit

Figure 6

Menu of Median Terms in Plexsys



```
Item      task priorities                   Matching   is_median_expression
                                            Expr.      statement_form_definition
                                            Found      task priorities
                                                       canonical_form
Input     statement_form_definition                    task priority
Expr.     task                                         aim
Match     aim                                          is realized by
                                                       task
                                                       priority

Matching Criteria: <A>ny Item Order
                   <R>elative
                   <S>trict    A

Item      Is used to assigned priorities to tasks RELATIVE TO the aims the tasks accomplish.
Comment

Expr.
Comment
```

Figure 7

Peruse Screen Format

Interfaces to Plexsys are batch and screen-oriented. A screen is a set of windows with full text editing capability. The language definition facility provided in Plexsys, LANGUAGE-EDITOR, is used to define forms. An example form definition is shown in Figure 3. The form definition includes the definition of language terms and language statements, instrumented as forms, as well as documentation on both terms and statement forms. The forms represent a nonprocedural language for target system description. The form definitions are stored in the JAMES database and are then used in target system descriptions. In Figure 4, the form defined in Figure 3 is used to define a specific instance of a task and its prioritized relation with an organization aim. Note that an aim is a general term given to all organizational objectives, goals, and strategies. Also, documentation on the language and target objects and statements can be stored. The documentation specified for the language definition serves as language documentation to target system modelers. The form definition tool supports editing of form definitions, performs consistency analysis for context-independent typings, and stores form definitions.

A tool, INSTANCE-EDITOR, is provided that allows the definition of target system descriptions using the forms defined with LANGUAGE-EDITOR. This generalized tool retrieves form definitions from the database, supports editing of target system descriptions using the forms, performs consistency checking on the target descriptions to insure consistent typing, and stores target descriptions. When invoked, INSTANCE-EDITOR traverses the JAMES database building selection menus of statement forms and language terms. Target sytem description input may be performed by selecting from the menu of specific statement forms or by selecting statement form groups based upon a term type. INSTANCE-EDITOR retrieves statement forms and language terms from the JAMES database. In Figure 5, INSTANCE-EDITOR has built a menu of all statement forms in the database from which the user can select specific forms to be filled out. Figure 6 shows the median term menu built by INSTANCE-EDITOR. Upon selecting a specific term, INSTANCE-EDITOR allows the user to cycle through the forms that concern or contain the term. For example, by selecting the statement group for "task," INSTANCE-EDIT will cycle through all statement forms that concern "task" allowing the user to fill in all information that concerns tasks.

A generalized database query facility and report generator, PERUSE, allows for interactive review of the current contents of the JAMES database and for genera-

PERUSE REPORT

This report summarizes the statement forms pertaining to aims.

```
aim priorities

   aim
   has sub
   aim
   priority

        aim priorities
           Is used to assign priorities to organizational aims.

        aim
           An "aim" is any organizational objective, goal, or strategy.
           An aim is, in essence, the purpose of organizational activities (tasks).

        has sub
           A relation to associate aims and their subaims in the aim hierarchy.

        priority
           Priority is a nine level ordinal scale used to prioritize
              1. Organizational aims,
              2. Tasks relative to the aims the help achieve, and
              3. Information sets relative to tasks using them.

           Note: The Plexsys priority analyzer is used to assess absolute priorities.

        task priorities
           aim
           is realized by
           priority

             task priorities
                Is used to assigned priorities to tasks RELATIVE TO the aims
                the tasks accomplish.

             is realized by
                A relation to associate aims and tasks.

             task
                A task is a business process or activity. Tasks are performed by agents
                at, perhaps, various locations. Information sets are supplied to tasks
                in support of their performance.
```

Figure 8

Peruse Generated Report

tion of customized reports. PERUSE supports the selection of statements in the JAMES database based on user specified terms and pattern matching criteria. As all model levels are stored in one JAMES database, PERUSE can be used to view the axiomatic terms, language definition and the target system descriptions. In Figure 7, the user is perusing all expressions in the database that contain the terms "statement form definition," "task," and "aim" in any permuted order. In effect then, the user is perusing all language form definitions that concern tasks *and* aims. The matching expression window contains the current statement form being viewed. The bottom windows contain the term, or item and expression documentation.

Figure 8 shows a report generated where the user has selected to report all language form definitions associated with "aim." Note that text may be entered at the terminal while using PERUSE such that the text is outputed to the report. In the example of Figure 8, this capability has been used to enter a report heading. Form definitions and the associated documentation serve as a "help facility" to modelers. Using this facility a modeler may ask to see all language forms that concern orgnizational tasks or that concern both tasks and information requirements, or generate a complete language manual, for example. A modeler may also generate documentation on the target system model.

Whereas consistency analysis of context-independent typing is performed before a language form or target system description is stored in the database, completeness checking is performed at user-controlled intervals during language definition and target system modeling. Completeness checking is supported by the generalized completeness analyzers. By using the case grammar paradigm discussed in the previous sections, completeness checking can be performed on the language definition itself. The pattern matching functions in JAMES allow for searching the median level for violations of the rules specified in the previous section. Pseudo-code for the algorithm for checking the number two median integrity rule is:

```
For each language term
    If the term is in the role of result
        If not
            find the term in the role of process
            or find the term in the role of actor
            or find the term in the role of instrument
            or find the term in the role of material
            or find the term in the role of directive
        Then 'incomplete language definition.'
```

Another analyzer reports violations of completeness rules for an arbitrary instantial model defined via an arbitrary median model or language definition. As noted,

```
Plexsys Version 0.1          3-JUL-1984 11:41:44.32
                 Sample Organization

  Incompleteness in user specified object: task

  task division budgeting
          is not defined in info set volume
                            task attributes
                            agent task
                            task location
                            task priority

  task regional sales evaluation
          is not defined in item currency accuracy
                            info set volume
                            info set response
                            info set time horizon
                            info set ordering

  task salesperson evaluation
          is not defined in item currency accuracy
                            info set volume
                            info set time horizon
                            info set access items
                            task attributes
                            agent task
                            task location

  task product evaluation
          is not defined in info set volume
                            info set time horizon
                            info set access items
                            agent task
                            task location
```

Figure 9

Generalized Instance Integrity Report for a User Specified Median Term

given that the median rules are satisfied by the language definition, integrity checking of the instantial model reduces to a simple, general algorithm.

```
For each instance object
    Find its corresponding median type
        For each statement form
            in which the median type appears
        If the instance object does not appear in
            an instance of a statement of that type
        Then report instance integrity violation.
```

Options provided for completeness checking of target systems descriptions include (1) global completeness checking, (2) completeness checking on all instances of a given language term, all "tasks" for example—see Figure

```
Plexsys Version 0.1          3-JUL-1984 11:53:55.53
                Sample Organization

  Incompleteness in user specified object: product
        evaluation

  task product evaluation
        is not defined in info set volume
                          info set time horizon
                          info set access items
                          task attributes
                          agent task
                          task location
```

Figure 10

Generalized Instance Integrity Report for a User Specified Instance Term

9—and (3) completeness checking for a given instance—see Figure 10.

Summary

A metasystem is comprised of three logical layers. The axiomatic level represents the highest level view of "what information systems are." The axiomatic model, and accompanying metasystem facilities, are used to define a descriptive language which, in turn, is used to describe target information systems. Since the axiomatic model embodied by a metasystem is the foundation upon which models are built, it is important that the axioms be robust. This robustness is defined in terms of scope, granularity, succinctness, and dynamism.

Existing metasystems, in their attempt to provide scope, have sacrificed granularity and succinctness. In adopting an overly abstracted axiomatic model, the existing systems fail to generalize rules of system integrity. Integrity at any level is a function of the integrity of upper levels, for example, an incomplete language definition implies the incompleteness of the instantial models defined using the language. It is desireable, therefore, not only to generalize integrity rules for instantial models but also for median models. By defining axiomatic terms that represent roles played by components in a system, a large number of integrity rules can be generalized. The roles and corresponding integrity rules can help insure the intra- and inter-level *semantic integrity* of both the language definition and the corresponding instance definition. This refinement to existing meta models makes it possible to perform generalized integrity checking on the language definition (median model) as well as the descriptions of target systems (instantial model). Further, given the Plexsys metasystem implementation, dynamism is offered. This dynamism allows for the modifications of model levels and automatic enforcement of the propagation of modifications to a given level to those levels below.

REFERENCES

Brachman, R.J. "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks," *IEEE Computer*, December, 1983, pp. 30-36.

Brodi, M.L. "Recent Issues in Database Specification," *ACM SIGMOD*, Volume 13, Number 3, April 1983, pp. 42-45.

Bruce, B. "Case Systems for Natural Language," *Artificial Intelligence*, Volume 6, 1975, pp. 327-360.

Cameron, R.D. and Ito, M.R. "Grammar Based Definition of Metaprogramming Systems," *ACM Transactions on Programming Languages and Systems*, Volume 6, Number 1, January 1984, 20-54.

Checkland, P. *Systems Thinking, Systems Practice*, John Wiley & Sons, New York, New York, 1981.

Chen, P.P. "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transactions on Data Base Systems*, Volume 1, Number 1, 1976.

Chomsky, N. "Deep Structure, Surface Structure, and Semantic Interpretation," in *Semantics*, D. Steinberg, and L. Jokobovits, (eds.) Cambridge University Press, Cambridge, Massachusetts, 1971, pp. 183-216.

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

Fillmore, C. "The Case for Case," in *Universals in Linguistic Theory*, Bach and Harms, (ed.) New York, Holt Rinehart, 1968.

Fillmore, C. "Types of Lexical Information," in *Semantics: An Interdisciplinary Reader*, Steinberg D. and Jakobovits, L. (eds.), Cambridge University Press, Cambridge, England 1971.

Halstead, M.H. *Elements of Software Science*, North-Holland Press, New York, New York, 1977.

Kang, K.C. *An Approach for Supporting System Development Methodologies for Developing a Complete and Consistent System Specification*, Ph.D. Thesis, University of Michigan, Michigan, 1982.

Kerschberg, L. Marchand, D., and Sen, A. "Information System Integration: A Metadata Management Approach," *Proceedings of the Fourth International Conference on Information Systems*, Ross, K. (ed.), Houston, Texas, December 1983.

Knuth, E., Rado, P., and Toth, A. "Preliminary Description of SDLA," Hungarian Academy of Sciences Working Paper, December 1979.

Konsynski, B.R. and Bracker, L.C. "Computer-Aided Analysis of Office Systems," *MIS Quarterly*, Volume 6, Number 1, pp. 1-17. March 1982.

Konsynski, B.R. and Bracker, W. "Defining Requirements for a Computer-Aided Network Design Package," *IEEE Data Communications*, July 1980, 75-84.

Konsynski, B.R. & Nunamaker, J.F. "Plexsys: A System Development System," in *Advanced System Development/Feasibility Techniques,* J.D. Couger, M.A. Colter, and R.W. Knapp, (eds.) John Wiley & Sons, New York, New York, 1982, pp. 399-424.

Kottemann, J.E. *Formalisms for Business Information Systems Development,* Ph. D. Dissertation, Department of Management Information Systems, University of Arizona, Tucson, Arizona, 1984.

Levene, A.A. and Mullery, G.P. "An Investigation of Requirement Specification Languages: Theory and Practice," *IEEE Computer,* May 1982, pp. 50-59.

Nadler, G. *The Planning and Design Approach,* John Wiley & Sons, New York, New York, 1981.

Nadler, G. Johnston, J. and Bailey, J. *Design Concepts for Information Systems,* American Institute of Industrial Engineers Inc., Atlanta, Georgia, 1975.

Schank, R.C. *Conceptual Information Processing,* North-Holland Press, New York, New York, 1975.

Siklossy, L. *Let's Talk Lisp,* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

Stott, J.W. *Principles for Computer-Aided Information Systems Development,* Ph. D. Dissertation, Department of Management Information Systems, University of Arizona, Tucson, Arizona, 1984.

Teichroew, D. and Hershey III, 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. 42-48.

Teichroew, D. and Gackowski "Checking A System Description in a PSA Data Base for Consistency and Completeness," ISDOS working paper 7742-0189-2, University of Michigan, June 1977.

Teichroew, D., Macasovic, P., Hershey III, E.A., and Yamamoto, Y. "Applications of the Entity-Relationship Approach to Information Processing System Modelling," in *ERA Approach to Systems Analysis and Design,* P Chen, (ed.) North-Holland Press, New York, New York, 1980 pp. 15-38.

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