DESIGN AND MODELING CONCEPTS

JERZY W. ROZENBLIT
BERNARD P. ZEIGLER
The University of Arizona
Tucson, Arizona

MULTI-OBJECTIVE SYSTEM MODELING

The ability to increase the decision-making capabilities in different environments is related to the scope of our possible intervention into the operation of the systems being modeled. Zeigler (1) classifies the levels of intervention into three broad categories: management, control, and design. Management type of intervention connotes determining policies whose interpretation and execution is then delegated to subordinate levels. Control intervention is an action deterministically related to policy. In contrast, design represents the greatest scope of intervention in that the designer either creates the "real system" or augments, modifies, or replaces a part of existing reality (see also ASSEMBLY, ROBOTIC, DESIGN FOR).

The effects of interventions are uncertain due to the existence of uncontrollable parts in the system and it becomes necessary to encode knowledge about such parts in models of the system. Models represent abstractions of the reality whose primary function is to capture the structural and behavioral relationships in the system. These relationships would be difficult to observe were the models not available.

Thus, the modeling methodology should be an inherent component in computer-aided decision systems in management, control, and design (1–4). The tools and activities prescribed by this methodology enable the decision makers to evaluate (based on the analysis of the models' simulation) the effects of interventions before they are actually carried out. The "best," in terms of performance measures related to the system under evaluation, intervention alternatives are chosen and finally deployed in the real system. The choice of performance measures reflects the objective the decision maker (be it an economist, a designer of a power plant, or a technician supervising a chemical process) attempts to achieve. The nature of the objectives orients and drives the modeling and simulation processes.

It is easy to conceive that any real system could be subject to a multiplicity of objectives in management, control, or design context. Consider an example to illustrate such a situation. Assume that a banking conglomerate is in the process of computerizing operations in several of its branches. Apart from having a computer network designed, the bank is establishing a Computing Services Department to handle and support the new system. Outlined below are some of the aspects relevant to the operation of the new computer system and department.

Objective

1. *Cost requirements (required model of)*: financing schemes and policies, investment capital, long-term maintenance costs, new personnel hiring and training, etc.
2. *Financial operations (required model of)*: network operation, likely speed-ups in transaction processing time and decreases in transaction processing costs, growth of service volume, interactions with other computerized institutions.
3. *Customer satisfaction (required model of)*: customer tastes (eg, popularity of dial-up access, automated teller machines), customer interface (waiting, accessibility, ease of completing a transaction, etc).
4. *Data security:* communication protocols, communication media (phone lines, hardware), security schemes.
5. *Personnel requirements:* system operation, human–machine interface.
6. *Quality control:* network operation, communication schemes, inter- and intrabranch interfaces.

It is rather unlikely that a comprehensive model of a computerized banking network, reflecting all of the above objectives, could be constructed. Even if it were possible to build such a model, its validation, and subsequently, verification and simulation would present a paramount degree of complexity. Instead, envision a collection of partial models, each reflecting a specific objective. This implies that the objectives orient the model building process by helping to demarcate the system boundaries and determine the model components of relevance (1,5). The fundamental formal concept supporting these activities is the *system entity structure*. The entity structure enables the modeler to encompass the boundaries and decompositions conceived for the system (1).

The role of the objectives is equally important in the process of specifying the experimentation aspects for the models that have been perceived for the real system. The key concept in this process is that of *experimental frame* ie, the specification of circumstances under which a model (or the real system) is to be observed and experimented with (1,6,7). The experimental frame definition reflects the objectives of modeling by subjecting the model to input stimuli (which in fact represent potential interventions), observing reactions of the model by collecting output data, and controlling the experimentation by placing relevant constraints on values of the designated model state variables. The data collected from such experiments serve as a means of evaluating the effects of intended interventions.

Generation of meaningful experimental conditions is not a trivial task and requires that the modeler understand the nature of the objectives and their interactions. A frame, similarly to a model, may reflect a single or a complex set of goals. Recall our example of the banking network and the following objectives: customer satisfaction, financial operation, and data security. When a comprehensive model (or partial models reflecting each aspect) is devloped, a relevant set of frames must be available in order to perform the experiment. Notice, however, that the above three objectives may conflict with one another. For example, improving customer satisfaction by providing dial-up access by public telephone lines might decrease the level of data security. On the other hand, purchase of an obsolete computer system may decrease customer satisfaction through the lack of convenient access and not provide the adequate level of security, even though the system might be suited to handle all of the bank's operations. Thus, meaningful trade-off experimental frames have to be specified, and the

models must be evaluated within the context of trade-off orderings over the set of objectives.

This article discusses and recognizes the multiplicity of objectives, models, and experimental frames as a *sine qua non* condition of the modern, advanced modeling methodologies. The focus is specifically on the issues concerning the system design, understood here as the use of modeling techniques to procure and evaluate a model of the system being designed. In the ensuing section, the synergism between simulation modeling and system design is underscored.

System Design and Modeling, Synergies

The design aspect in decision-making offers the widest scope of intervention in that the designer develops a model from which a new system will be created. As opposed to system analysis, where the model is derived from an existing, real system, in system design the model comes first as a set of "blueprints" from which the system will be built, implemented, or deployed (8,9). The blueprints might take several forms; they could be simple verbal informal descriptions, a set of equations, or a complex computer program. The goal of such defined system design is to study models of designs before they are actually implemented and physically realized.

At this point, the fundamental question arises: Why are modeling and simulation methodologies needed to support system design? Before this question is answered, the basic elements in the dynamics of the design process must be summarized (9–13):

1. Designs are created by individuals who use the basic problem-solving techniques, namely, problem definition, proposal of a solution, and test of the solution against the problem definition.
2. The problems being addressed are often of a large scale. Thus, there should be methods for decomposing the problems into subproblems easily comprehensible by the designer. Partial solutions could then be generated and integrated using proper aggregation mechanisms.
3. Solutions (designs) are built based on the designer's perception of reality; they result from the transformation of the designer's ideas and knowledge into a blueprint of the system to be created.
4. The attributes of design should be described in comparative measures and applied by using trade-off techniques.
5. The tools, techniques, and methods are currently mostly manual methodologies; automated tools for system design are only now evolving (14,15).

Recall that the primary goal is to locate the system design within the modeling framework. An attempt is made to provide a systematic methodology for a design process supported by adequate formal structures and leading toward future computerization. As depicted in Figure 1, system design is brought into the multifaceted framework with design process being supported by the modeling and simulation techniques in the manner described below.

Modeling is a creative act of individuals using the basic problem-solving techniques, building conceptual models based on the knowledge and perception of reality, requirements, and objectives of the modeling project. The models are design blueprints. This constitutes direct relation to points 1 and 3.

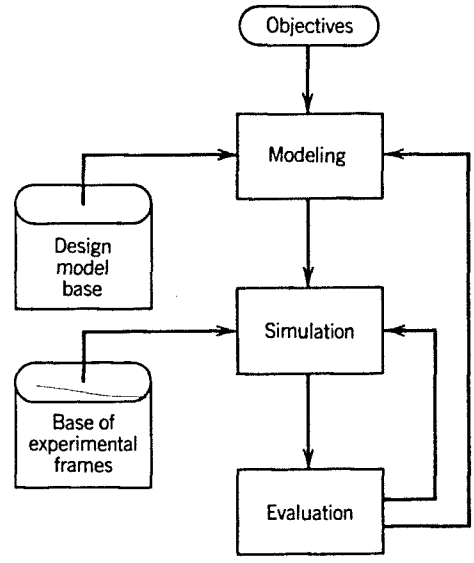


Figure 1. Design in the multifaceted modeling context.

By providing mechanisms for model decomposition, hierarchical specification, and aggregation of partial models (1), the multifaceted modeling approach fully responds to the needs of system design signaled in point 2. Adding the previously mentioned system entity structure, one is now equipped with a facility to generate families of models (of design) in various decomposition aspects.

The experimental frame concept responds to the needs of point 4. This unique structure provides a spectrum of performance evaluation methods, including evaluation of multilevel, multicomponent, hierarchically specified models (13).

Finally, the underlying purpose of the multifacted modeling is that it provides structures implementable in computerized support environments. This is where the possibility for a response to the drastically growing needs for compter-aided design tools is envisioned. The current tools lack an underlying theoretical framework that permits a uniform treatment of system design by providing concepts such as structure and behavior, decomposition, and hierarchy of specification (1,15). The existing architectures are usually conglomerates of various, often incompatible, tools whose coordination poses serious problems and often defies their purpose (4,14,16,17). The representation schemes offered by the multifacted framework are well structured and have formalized operations that can exploit such structures. This significantly reduces the effort of designing expert computer-based environments.

In the following sections, a conceptual framework for model-based design is set up. The proposed methodology utilizes the formal modeling concepts. Several major steps underlie the methodology:

- The system entity structure is a basic means of organizing a family of possible configurations of the system being designed.

- The objectives and requirements of the design project induce appropriate generic experimental frames.
- The design entity structure is pruned with respect to the generic frames. This results in a family of design configurations that conforms to the design objectives.
- The pruned substructures serve as skeletons for generating rules for synthesis of design models.
- Resulting models are evaluated in respective experimental frames and the best design models are chosen on the basis of such evaluations.

FORMAL FRAMEWORK FOR MODEL-BASED SYSTEM DESIGN

This section provides the necessary formal background for the multifaceted system design introduced earlier.

The System Entity Structure

To represent a family of design configurations appropriately, a structure is needed that embodies knowledge about the following relationships: decomposition, taxonomy, and coupling. The *decomposition* scheme allows a representation in which an object (component of a system being designed) is decomposed into components. The structure should be able to operate on and communicate about the decomposition scheme.

Taxonomy is a representation for the kinds of variants that are possible for an object, ie, how they can be categorized and subclassified.

The third kind of knowledge to represent is that of *coupling constraints* on the possible ways in which components identified in decompositions can be coupled together.

A formal object that embodies these three basic relationships is called the *system entity structure.* The system entity structure is based on a treelike graph encompassing the system boundaries and decompositions that have been conceived for the system. An *entity* signifies a conceptual part of the system that has been identified as a component in one or more decompositions. Each such decomposition is called an *aspect.* Thus, entities and aspects should be thought of as components and decompositions, respectively. The system entity structure organizes possibilities for a variety of system decompositions and model constructions.

Both entities and aspects can have attributes represented by the so-called *attached variables* types. When a variable type V is attached to an item occurrence I, this signifies that a variable I.V may be used to describe the item occurrence I. Therefore, whereas an unqualified variable type such as LENGTH may have multiple occurrences in the entity structure, a qualified variable, eg, QUEUE1, LENGTH belongs to one and only one item occurrence, QUEUE1.

The system entity structure satisfies the following axioms (18,19):

1. *Uniformity:* any two nodes that have the same labels, have identical variable types, and isomorphic substructures.
2. *Strict hierarchy:* no label appears more than once down any path of the tree.
3. *Alternating mode:* each node has a mode that is either "entity" or "aspect" (decomposition or specialization); the mode of a node and the modes of its successors are always opposites. The mode of the root is entity.
4. *Valid brothers:* no two brothers have the same label.
5. *Attached variables:* no two variables attached to the same item have the same type.

The entity/aspect distinction can be interpreted as follows: an entity represents an object of the system being designed, which either can be independently identified or is postulated as a component in some decomposition of the system. An aspect represents one decomposition, out of many possible, of an entity. The entities of an aspect represent disjoint components of a decomposition induced by the aspect. The aspects of an entity do not necessarily represent disjoint decompositions.

For a more detailed and formal treatment of the entity structure concept, consult Ref. 1. How the discussed knowledge representation scheme is realized by the concept is discussed below.

An entity may have several specializations. Each specialization may in turn have several entities. The original entity is called a general type relative to the entities belonging to a specialization, which are called special types. Since each such entity may have several specilizations, a *hierarchical structure* results which is called a *taxonomy* (1,20,21).

The alteration property is a salient feature which requires that entities and specializations alternate along any path from the root to leaves. The same property holds for entities and aspects. Specializations have independent existence, just as entities do. A specialization may occur in more than one location. Whenever it occurs, it carries with it all its attributes and substructures. Of course, it may not be meaningful to attach a particular specialization to a particular entity.

Hierarchical decomposition is in many ways analogous to the specialization hierarchy just discussed. The alternation property now requires alternation of aspects and entities. An aspect is a mode of decomposition for an entity just as a specialization is mode of classification for it. There may be several ways of decomposing an object, just as there may be several ways of classifying it. Formally, aspects and specializations are quite alike in their behavior. They each alternate with entities but cannot be hung from each other. A special type of decomposition called a *multiple* decomposition facilitates flexible representation of multiple entities whose number is in a system may vary. (Throughout the illustrations, the multiple decomposition aspect is denoted by triple vertical bars and specializations by double vertical bars.)

Specialization is a concept distinct from that of decomposition. However, there is a way of mapping a specialization hierarchy into an equivalent decomposition aspect which intimately involves the multiple decomposition concept. The transformation, illustrated in Figure 2, is simple: if entities A1 and A2 are special types of entity A, then the multiple component As is decomposable into the multiple components A1s and A2s.

To express the coupling constraints, the following procedure is employed: apply the mapping to remove the specializations to obtain an entity structure containing only entities and as-

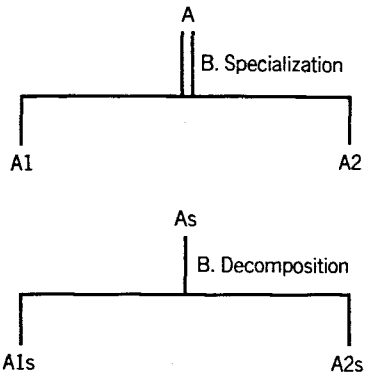


Figure 2. Removing specialization using the decomposition aspect.

pects. Now imagine that models are synthesized by working down the entity structure, selecting a single aspect for each entity and zero or more entities for each aspect. Such a process is called *pruning* of the entity structure (see Entity Structure and Experimental Frame-based Design Model Development). A pruning procedure is also defined as one that operates directly on entity structures with specializations. The coupling constraints must then be associated with aspects, since they represent the decompositions chosen when pruning. Moreover, a constraint must be associated with an aspect that contains all the entities involved in that constraint. What is more, this aspect should be minimal in the sense that there be no other aspect that lies below it in the entity structure which also encompasses all the entities involved in the constraint.

The following example illustrates how the system entity structure can be employed as a representation scheme for a family of design possibilities.

Assume that an aerospace agency is planning to launch a fully automated space station. In the first stages of development, the entity structures representing various configurations for the stations are proposed. One of the possible design entity structures is depicted in Figure 3.

For the sake of brevity, only three aspects, the automation, control, and communications aspects, are presented in the illustration. First, the automation aspect: assume that one of the design objectives is that the station be capable of performing a number of different tasks using a coordinated group of robots (subsequently called robot organization). The tasks, eg, station keeping, maintenance, launching satellites, refueling other space ships, may be performed at different sites called workstations.

The organization, as shown in Figure 3, may employ various types of robots. The two specialized types present in the structure are termed executive and functional robots. It is assumed that an executive robot has managerial skills, (ie, it can coordinate, hire, and fire robots in the task accomplishment process. A functional robot can be coordinated by an executive one and perform tasks for which it has been designed and programmed.

The robots, viewed from the standpoint of their organizational structure, are specialized into adaptive and nonadaptive organizations. The adaptive architecture is further specialized into a hierarchical tree-based architecture and a nonhierarchical distributed adaptive organization. In the tree-based architecture, the robots are recruited from the availability pool and returned to it according to the adaptive reconfiguration strategy. The above scheme, termed "hire/fire," has been proposed by Zeigler and Reynolds (22,23), who are investigating adaptive computer architectures.

The nonhierarchical organization can be visualized as a parallel-serial type of organization whose adaptation scheme may be based on the schemes similar to that of a CPM method. Assuming that a task can be represented by an activity net-

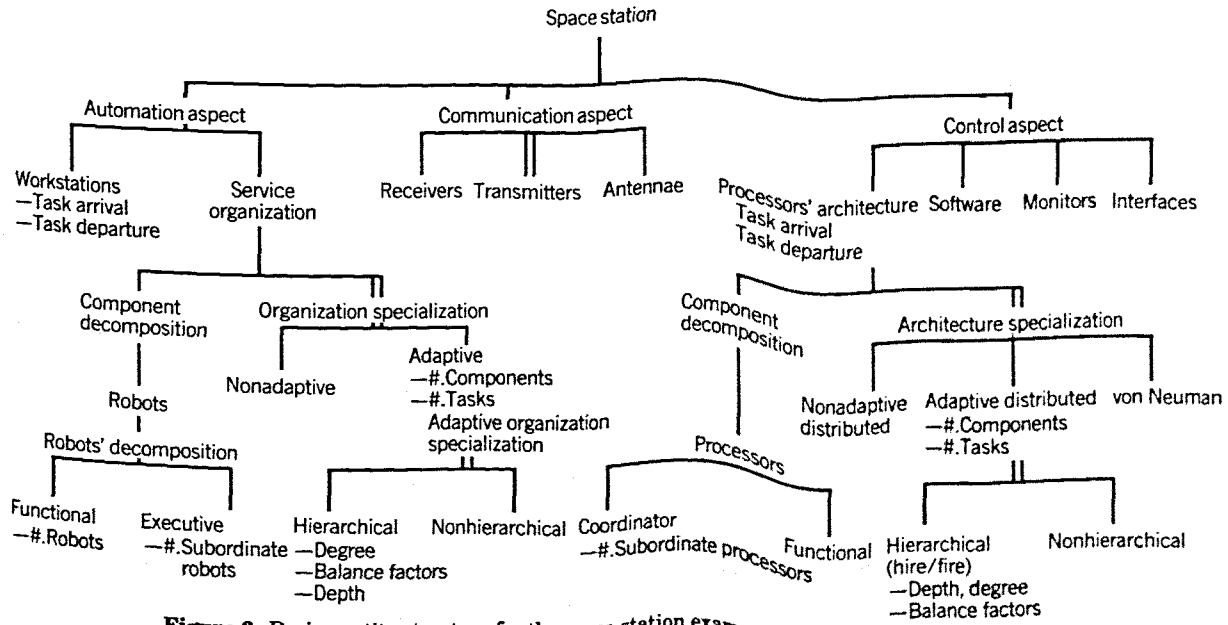


Figure 3. Design entity structure for the space station example. –, denotes an attached variable type.

work, the adaptation consists in shifting the robots from the least to the most critical activities.

In the control aspect, the processors, software, and real-time monitors are the basic components of the station. The processors, like the robots, may have the following architecture specialization types: classic von Neuman, distributed nonadaptive, and an adaptive hire/fire organization. The adaptive architecture requires that functional processors (equivalent in their role to functional robots) and coordinators (equivalent to executive robots) be present.

The Experimental Frame Definition

The system entity concept facilitates the representation of design structures. A means for expressing the dynamics of the design models is also needed. Since the design framework is objectives-driven, the experimental frame concept is used as the other underlying object in system design. The role of experimental frames will be twofold. First, the frame will represent the behavioral aspects of the design objectives and facilitate retrieval of entity structures that conform to those objectives. Secondly, the experimental frame will serve as a means of evaluating the design models with respect to given performance measures.

The conceptual basis for a methodology of model construction in which the objectives of modeling play the key and formally recognized role (therefore called *objectives-driven methodology*) was laid down by Zeigler (1).

The basic process in this methodology is that of defining an experimental frame, ie, a set of circumstances under which a model or real system is to be observed and experimented with. This process comprises the following steps. The purposes (objectives) for which the simulation study is undertaken lead to asking specific questions about the system to be simulated. This in turn requires that appropriate variables be defined so that a modeler can answer these questions. Ultimately, such a choice of variables is reflected in experimental frames that also express constraints on the trajectories of the chosen variables. The constraints on observations and control of an experiment should be in agreement with the modeling objectives. A choice of relevant variables constitutes the first important stage of experimental frame specification. The next step is to categorize the variables into input, output, and run control and place constraints on the time segments of these variables. Formally, the experimental frame specifies the following seven tuple:

$$EF = <T,I,O,C,W_I,W_C,SU,W_{SU}>$$

where

- T is a time base
- I is the set of *input variables*
- O is the set of *output variables*
- C is the set of *run control variables*
- W_I is the set of *admissible input segments*, ie, a subset of all time segments over the cross product of the input variable ranges
- W_C is the set of *run control segments*, ie, a subset of all time segments over the cross product of the control variable range.
- SU is a set of summary variables
- W_{SU} = {s:s:I × O → SU.range} is the set of *summary mappings*

The I/O data space defined by the frame is the set of all pairs of I/O segments:

$$D = \{(w,r): w\epsilon (T,X), r\epsilon (T,Y) \text{ and } dom(w) = (r)\}$$

where X and Y are input and output value sets, respectively.

Since experimental frames should have a meaningful interpretation for both the model and the real system, a concept of restricting the initial state for the model must be provided. The run control variables serve this purpose. They initialize the experiments and set up conditions for their continuation and termination. The set of initial values for the run control variables is called INITIAL. The subset of the control space defined by the termination conditions is called TERMINAL. The set of run control segments is then defined as (for a detailed formal treatment of the experimental frame concept, see Ref. 1):

$$W_C = \{m:m:<t_i,t_f> \rightarrow Z$$
$$\text{and } m(t_i)\epsilon \text{ INITIAL, } m(t)\epsilon \text{ CONTINUATION for } t\epsilon\{(t_i,t_f)\}$$

where Z = cross product of the ranges of individual control variables, and t_i and t_f are the beginning and end of the observation interval, respectively.

CHARACTERIZATION OF THE MULTILEVEL, MULTIPHASE SYSTEM DESIGN

System design concepts are found in many disciplines. The paradigms of each discipline underlie the methods for design representation and methodology. In systems theory, the dominant framework is the mathematical representation. In systems methodology, the methods are adopted from operations modeling; in philosophy, the models of thinking play an important role. A wide spectrum of studies on the subject has been documented in the literature (8,9,11,12,24–27).

Common Traits in System Design Methodologies

Reviewing most of the conventional design methodologies leads to the following scheme of reasoning (8,9,12,24,27):

1. State the problem.
2. Identify goals and objectives.
3. Generate alternative solutions.
4. Develop a model.
5. Evaluate the alternatives.
6. Implement the results.

The methods to address each of the above aspects of design depend on the discipline and often vary in the degree that they are found in most of the approaches. The most important traits are

1. *Objectives-driven nature of design.* Everything in the designed system is considered and evaluated in relation to the purposes of the project.
2. *Hierarchical nature of design.* Structures of systems being designed are represented in multilevel hierarchies that express decompositions of the system into subsystems (28).

3. *Design as decision making.* Design is concerned with actions to be taken in the future. Thus, incorrect decisions in the early stages of design may impede all subsequent actions.
4. *Iterative nature of design.* It is widely recognized that the design process should be iterative in that the designer should be able to return to earlier phases of design from any stage of the process. Analogies are often made here with cybernetics and control theory where iteration is continued until a desired equilibrium point is reached (12,28).
5. *Optimization in design.* Design seeks optimization of the whole system with respect to its objectives. Careful consideration must be given if attempts are made to optimize subsystems separately. Appropriate coordination methods must then be used to attain the overall objective (28).

Orthogonal System Design

In the context of this article the term system design will denote the use of modeling and simulation techniques to build and evaluate models of the system being designed.

The design process is considered a series of successive refinements comprising two types of activities. The first type concerns the specification of design levels in a hierarchical manner. The design levels are successive refinements of the decomposition of the system under consideration. The first, and thus the most abstract level, is defined by the behavioral description of the system. Subsequently, the next levels are defined by decomposing the system into subsystems (modules, components) and applying decompositions to such subsystems until the resulting components are not further decomposable. The atomic system components are represented at the lowest level of the design hierarchy. At each level, the specialization of components into different categories is allowed for. This facilitates the representation of design alternatives.

The second type of design activity is concerned with "horizontal" actions associated with design levels. Such actions include requirements specification, system functional description, development of design models, experimentation by simulation, evaluation of results, and choice of the best design solution.

The design should proceed along both axes of the above characterization. The designer must be able to structure the designs, explore alternative structures, and derive complete specifications and models at any level. Transitions between design levels must be possible and easy to perform.

Such an orthogonal specification of the design process is often called a design matrix. The design process is represented in Figure 4. The space defined by the cross product of phases and levels is in various methodologies filled with different methods, techniques, and tools. However, such methods and tools are often incompatible and there are no underlying structures integrating the design steps at various levels and phases. This, of course, motivates and justifies efforts to solve that problem by applying structures such as the system entity formalism and experimental frame.

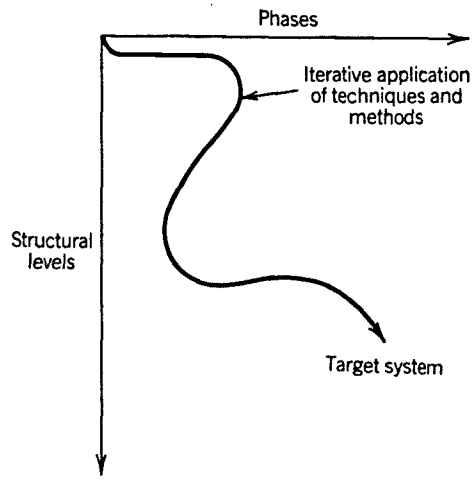


Figure 4. Orthogonal system design.

To illustrate how the levels of design can be defined, consider the class of computer system designs (16,17,29).

It is common view in the literature (14,16) that the design levels for computer system design are based on the following scheme:

1. *Behavioral level:* general description of system behavior, purpose, and attributes.
2. *Functional system architecture level:* functional system characteristics, partitioning of the system into hardware/software functional specifications.
3. *Hardware/software architecture level:* representation of the system in terms of the main building blocks, (eg, processors, memory, peripherals, etc.)
4. *Module level:* detailed description of the architecture-level building blocks.
5. *Register-transfer level:* the data paths, control sequences, and timing diagrams that implement the modules.
6. *Logic level:* specification of the system in terms of the detailed interconnection of logic components.
7. *Circuit level:* representation by physical circuits.

The phases for this type of design are the horizontal activities that have been specified above.

Although an attempt to refine the definition of design further is not made, nor a description of all its phases, how the multifaceted modeling with its formal objects can unify the design activities and support the construction of environments for expert system design is shown.

ENTITY STRUCTURE AND EXPERIMENTAL FRAME-BASED DESIGN MODEL DEVELOPMENT

This section presents a framework that operationalizes the system entity structure and the experimental frame concept into a systematic design framework. First, the choice of the system entity structure as the underlying object in our design methodology is justified.

Recall that the entity structure represents the following relations:

Decomposition hierarchy of the system being designed. This enables the direct representation of the design levels as discussed earlier.

Taxonomy. This relationship (captured by the specialization aspects) facilitates the classification of design components and constitutes a means of expressing various design alternatives.

Coupling constraints on the possible ways in which components identified in decompositions/specializations can be coupled. This enables the use of the system entity structure as a basis for the hierarchical design model construction.

Again, it is emphasized that the entity structure is the basic means for organizing the family of possible design configurations (structures, architectures). It also serves as a skeleton for the hierarchical design model construction.

It is equally important to understand the role of the experimental frame concept in the design process. The role of frames is twofold. First, an experimental frame is a means of representing the performance measures associated with the behavioral aspects of design objectives. Subsequently, the frame is used in a simulation experiment performed to evaluate the merits of a design model. There is, however, a second important role of the frame concept. A generic form of an experimental frame is employed in the design framework to delimit the design model space given by the system entity structure.

Generic Experimental Frames

A generic experimental frame type represents a general class from which experimental frame specifications can be derived. A generic frame is defined by unqualified generic variable types that correspond to the objectives for which the design study is undertaken.

Design objectives are associated with performance indexes that allow for a final judgment of the design models. A generic frame GEF is defined as the following structure induced by the performance index pi:

$$GEF_{pi} = \{IG, OG, W_{GI}, SU, W_{SU}\}$$

where GEF_{pi} denotes a generic experimental frame for performance index pi and

IG is the set of generic input variable types for pi
OG is the set of generic output variable types for pi
W_{IG} is the set of generic input segment types for pi
SU is the set of summary variables
W_{SU} is the set of standard summary mappings

To illustrate how a generic experimental frame can be defined, let us return to the space station example discussed earlier. Suppose a frame is to be defined that represents the questions concerning the measures associated with the adaptive robot (or computer) architectures perceived for the station (see Fig. 3).

There are two basic measures to be considered. The first one is concerned with the performance measures that are task-oriented, eg, throughput, task transit time, volume of tasks in the system, etc. The other concerns the organization itself, ie, to collect data on the structural properties of the organization.

The following generic frame for the evaluation of tree-based adaptive organizations is proposed:

Generic Frame: Adaptability

{This template is intended to generate experimental frame for collecting observations in adaptive, tree-based organizations}

Generic Input Variables

Task.arrival with range {task identifiers, task type}

Generic Input Segment

Task.workload—a discrete event segment with varying inter-arrival rate (eg, high, low) and/or varying task types
{One of the objectives of the frame is to observe adaptation patterns to changes in the input segment}

Generic Output Variables

Task.departure with range task identifiers, task types
Number of tasks being processed
Total number of coordinator-type components in the adaptive organization at any given time
Total number of functional-type components in the adaptive organization at any given time
Depth of the tree at any given time
Degree of the tree at any given time
Balance factors (for any or all the nodes)
Number of subordinates for coordinator-type components

Summary Variables

Departure/arrival rate ratio
Average task.transit time
Total number of tasks processed over the observation interval
Average number of components of the organization
Average number of coordinators
Average number of functional components
Number of coordinators/functional components ratio
Average depth of the tree
Maximum depth of the organization tree
Minimum depth of the organization tree

Other examples of generic experimental frame types for various performance criteria are presented in Ref. 30.

The Entity Structure-based Generation of Design Model Structures

Due to the multiplicity of aspects and specializations, the design entity structure offers a spectrum of design alternatives. The procedures that limit the set of design configurations by extracting only those substructures that conform to the design objectives are presented now. Called pruning, this extraction process is based on the following scheme. Assume that an entity structure has been transformed into a structure with no specializations. Then, imagine that the structure is tra-

versed by selecting a single aspect for each entity and zero or more entities for each aspect. All selected entities carry their attributes with them. Also, the coupling constraint of the selected aspect is attached to the entity to which this aspect belongs.

The above process results in *decomposition trees* (1) that represent hierarchical decompositions of design models into components (*design model structures*). The process that extracts the model structures from the design entity structure is called *pruning*.

By pruning the system entity structure with respect to generic frames, the following benefits are obtained:

1. In terms of the contribution to the design process:
   a. a generic frame extracts only those substructures that conform to the design objectives. Thus, a number of design alternatives may be disregarded as not applicable or not realizable for a given problem.
   b. partial models of the design can be formulated and evaluated. This may significantly reduce the complexity which would arise dealing with the overall design model. The generic frame may thus be viewed as an object that partitions the system entity structure into design-objective related classes.
   c. The evaluation of design models constructed from the pruned substructures is performed in corresponding experimental frames. Such frames are generated by instantiating the generic frames used to prune the system entity structure. Hence, automatic evaluation procedures could be employed in the design process.
2. In terms of facilitating the pruning process itself, generic frames automatically determine:
   a. The aspects that are selected for each entity.
   b. The depth of the pruning process.
   c. The descriptive variables of components.

Pruning Algorithms This section presents a suite of pruning algorithms for generating design model structures and begins with the definition of the procedure Prune for pruning pure entity structures, ie, structures in which no specialization relations occur. For entity structures in which specializations are present, procedures for mapping such structures into a set of pure entity structures are provided. The procedure Prune is then applied to the set of pure structures, and relevant design model structures are generated as a result of pruning.

In presenting the algorithms, the concept of a nondeterministic algorithm is employed, ie, the algorithms are allowed to contain operations whose outcome is not uniquely defined but limited to a specified set of possibilities. In the case of pruning pure entity structures, the purpose of the nondeterministic version of the algorithm is to provide a definition for a set of all structures that the deterministic version should produce to be correct.

The case of pruning the specialized entity structures is more complex. The choice of a nondeterministic algorithm is justified in that there is no deterministic procedure that generates the solution in polynomial time.

For the pruning process, it is enough to restrict the generic experimental frame to the generic observation frame (5) ie,

$$GOF = \{IG, OG\}$$

where IG denotes the set of generic input variable types and OG the set of generic output variable types. By defining the observation frame as above, its role is restricted to representing *behavioral* aspects of design objectives. There are also objectives that constrain the structural aspects of the project under consideration. Therefore, in order to realize the strutural constraints, it will be necessary to augment the design model development with a process termed *synthesis rule generation*.

Return to the pruning procedure and define how a generic observation frame generates the design model structures that accommodate behavioral design objectives. Given the generic observation frame, all the substructures that have all the input and output variable types present in that frame are extracted.

First, a nondeterministic version of such a procedure is defined. Assume that the function *choose* selects an aspect for an entity. The algorithm presented in Figure 5 returns a nondeterministically selected decomposition tree that accommodates the generic frame in which the pruning proceeds.

The deterministic version of procedure Prune is based on the depth first tree traversal. In this procedure, every entity in each aspect is searched for occurrences of variable types that are present in the generic observation frame. The entities are attached to the model decomposition tree as the search progresses. At the same time, the algorithm calls itself recursively for each entity being searched. The complete deterministic pruning procedure is given in Figure 6.

To initialize the pruning process, the steps given below are followed:

1. In the system entity structure, choose the entity E_i that represents the model to be evaluated (this entity will label the root of the model structure TE_i).
2. Create a dummy entity DE (with no variables) with a dummy aspect DA in which E_i is a subentity of DE.
3. Call Prune (DE, CV_{GOF}, V_{GOF});

After the procedure has been executed, DE must be eliminated from all the model structures.

The procedure Prune generates a set of design model structures in the form of decomposition trees. Each such structure accommodates the generic observation frame GOF and constitutes a skeleton for a hierarchical model construction. Figure 7 illustrates the results of pruning of the system entity structure with respect to frame GOF.

The limitation of the procedure is that it operates only on pure entity structures, ie, those that do not have specializations. One possible way of dealing with the problem is to map the entity structure that contains specializations into a pure structure using the multiple decomposition concept and the transformation discussed in System Design and Modeling, Synergies. Then the procedure Prune can be applied to the pure structure. However, this approach greatly increases the size of the entity structure and forces the use of multiple entities. This may not be well justified in some design problems. Therefore, the Prune procedure is extended to operate on entity structures that contain specializations.

For the new algorithm, first it is assumed that if there is

```
Nondeterministic Prune(E_j, CV_GOF, V_GOF);
E_j – root of the pure entity structure
CV_GOF – set of variables of the generic frame GOF
          this set is used to check if all the frame variables
          are present in the pruned substructure initially
          CV_GOF = V_GOF
V_GOF set of input and output variable types of GOF
 failure – signals an unsuccessful completion
 success – signals a successful completion

begin
 A_i := choose(E_j);
 for each E_k ∈ A_i do

  begin
   attach E_k with all its variables as a child of TE_j;
   attach coupling constraint of aspect A_i to TE_j;
   { TE_j denotes the node corresponding to E_j in the model
   structure }
   Prune(E_k, CV_GOF, V_GOF);
  end;


     for each node in decomposition tree TE_j do
       begin { verify correctness of choice }
         visit node;
         if v_k ∈ V_GOF then CV_GOF := CV_GOF – v_k;
         { the attached variable types v_k that belong to the
         generic frame GOF are marked as have been found in
         the model structure }
       end; { of for each node }
      if CV_GOF is not empty then failure
        { the substructure does not have all the variables present
        in the frame }

      output Decomposition Tree TE_j;
      { the tree TE_j is a model structure that accommodates the
        frame GOF }
      success;

end. { of Nondeterministic Prune }
```

Figure 5. Nondeterministic prune algorithm.

more than one specialization at any given level in the entity structure, such specializations are hung from one another to reduce their number to one at this level. This operation is illustrated in Figure 8.

Secondly, if a general entity that has a specialization is a component in a multiple decomposition of a multiple entity, then the specialization is mapped into the decomposition aspect according to the rules presented earlier. Given these assumptions the extended pruning process is defined.

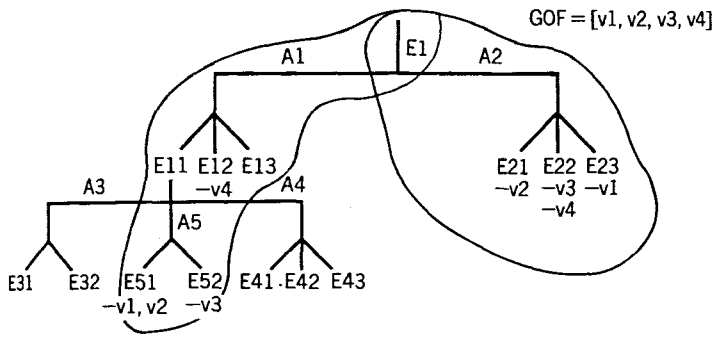
As depicted in Figure 9, the extended procedure employs three basic modules. The Move module removes all aspects from each level of the design entity structures where specializations occur. The aspects with their full substructures are hung from the entities of the specialization. This is in agreement with the inheritance property that states that the specialized entities inherit the attributes and substructures of the general entity. The algorithm that performs this function is illustrated in Figure 10.

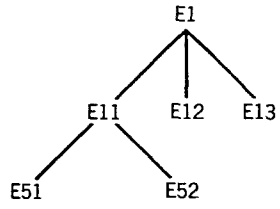
The Move module returns an entity structure that at any given level has either aspects or a specialization. Given such an entity structure, a set of pure entity structures is generated

```
Procedure Prune(E_j, CV_GOF, V_GOF); {Deterministic}
{ This procedure prunes the pure system entity structure and
 returns the model structures that accommodate the generic
 observation frame GOF. Multiple occurrences of a frame
 variable type are permitted in the model structures }

begin
 for each aspect A_i ∈ E_j do
  begin

   for each entity E_k ∈ A_i do
    begin

     attach E_k with all its variables as a child of TE_j;
     { TE_j denotes the root of the model structure being
     built }

     CV_GOF :=CV_GOF – v_k;

     { update the current set CV_GOF by subtracting the
      variable types v_k such that v_k ∈ V_GOF and v_k is
      attached to E_k }

     if E_k has at least one variable type present in V_GOF
     then mark this level in the model structure as the last
          level at which variable types present in the
          frame have been found;

    end; { of for each entity . . . }

   attach the coupling constraint of the aspect A_i to TE_j;

   for each E_k ∈ A_i such that E_k has aspects do
    Prune (E_k, CV_GOF, V_GOF);
    if C_GOF is empty { ie, the frame is accommodated }
    then
      begin

        create a copy of the current model structure rooted
        by TE_j;

        { this copy will serve as a basis for model structure
        construction in the next aspect A_i+1 }

        output the current model structure rooted by TE_j
        without the entities that appear below the level
        marked as the last level with frame variable type
        occurrence;

      end; { of if }

    update the current structure TE_j by cutting off the last
    level entities;

    { thus prepare the structure for pruning in the next
     aspect }

   end; { of for each aspect . . . }

end. {of prune }
```

Figure 6. Deterministic prune algorithm.

by substituting the specialized entities in place of the general ones. This process is called Split. Notice that a deterministic algorithm to perform the split would have to generate all the combinations of pure entity structures resulting from substi-





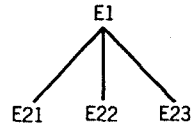


Figure 7. Pruning of the system entity structure in the generic frame type GOF and resulting structures for model construction.

tuting the specialized entities into the general ones. Thus, the problem is computationally hard. Instead, a nondeterministic algorithm that employs the *choose* function for selecting one specialized entity from a set of entities in a given specialization relation is proposed (Fig. 11).

The last module, Prune-Set, employs the deterministic procedure Prune (see Fig. 12).

To illustrate how the extended pruning process works, consider a substructure of the space station design entity structure, depicted in Figure 13. This substructure has both aspects and specializations. The components aspect of the entity Service Organization is first moved down and attached as an aspect to the specialized entities Adaptive and Nonadaptive Organization. This results in a structure shown in Figure 14. The same process is applied again to the specialized entities of Adaptive Organization; the result is depicted in Figure 15. Consequently, an entity structure in which at any level there are either specializations or aspects (never both) is obtained. Finally, the structure is split into pure entity structures with the specialized entities in place of the general ones.

The extended pruning with respect to the generic frame "Adaptability," applied to the station design entity structure, results in the model structures depicted in Figures 16a and 16b, respectively.

DESIGN MODEL SYNTHESIS

The pruning process described in the foregoing section restricts the space of possibilities for selection of components and cou-

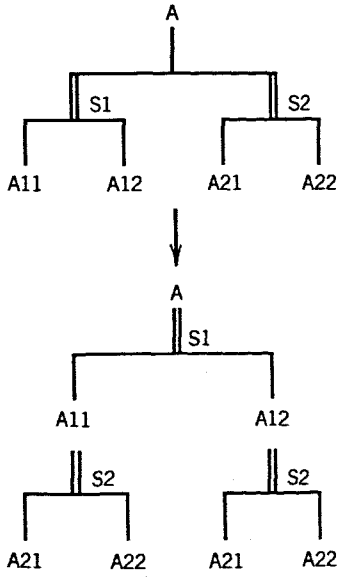


Figure 8. Preparing the entity structure with specializations for pruning.

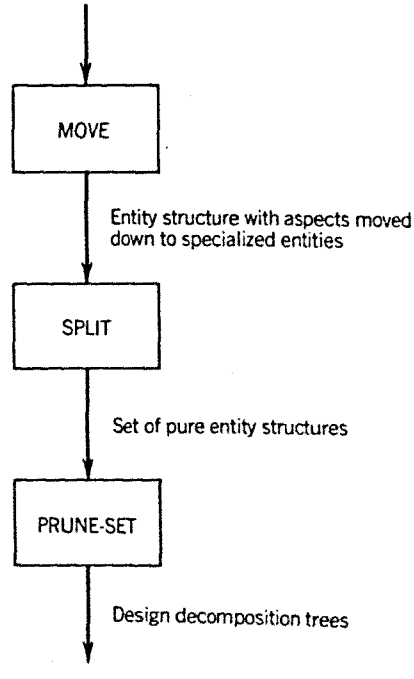


Figure 9. Diagram of extended pruning procedure.

```
Procedure Move(E_j);
    { this module moves all the aspects and variables down to
    the specialized entities }

begin
  if E_j has a specialization S

   begin
      for each aspect A_1 ∈ E_j do

        begin
         for each entity E_k ∈ S do

          begin
            copy all the variables of E_j to E_k;
            attach A_1 with its full substructure and variables as
            an aspect of E_k;
           end;
          delete A_1 from aspects of E_j;
         end;

      for each entity E_k ∈ S do
        Move(E_k);
   end
  else
    begin
      for each aspect A_i ∈ E_j
        for each entity E_k ∈ A_i
         call Move(E_k)
    end;
end. { of Move }
```

Figure 10. The Move algorithm.

plings that can be used to realize the system being designed. Thus, it is assumed that the design can now be equivalent to the synthesis of a design model based on the pruned structures and the structural constraints imposed by the project require-

```
Nondeterministic Split(E_j, ES);
  ES – entity structure with specializations

begin
 if E_j has aspects then
  begin
    for each aspect A_i ∈ E_j do
      for each entity E_k ∈ A_i do
          Split(E_k, ES);

 end
else
 if E_j has a specialization then
  begin
    choose(E_i) { an entity in this specialization };
    replace E_j with E_i;
    cut off all the other entities in this
    specialization (including their substructures)
    from the system entity structure ES;
    Split(E_i, ES);

  end;
end. { of Split }
```

Figure 11. Nondeterministic algorithm split.

```
Procedure Prune—Set({E_j}, CV_GOF, V_GOF);
    { this procedure prunes all the pure entity structures that
     result from algorithm Split }

begin
    for all the pure structures E_j do
      call Prune(E_j, CV_GOF, V_GOF);
      { invoke the procedure Prune to generate a
       set of design decomposition trees }
end. { of Prune—Set }
```

Figure 12. Algorithm Prune_Set.

ments. The following synthesis rule development methodology is proposed:

- Restrict the design domain by pruning the design entity structure in respective generic observation frames.
- Examine the resulting substructures and their constraints. Try to convert as many constraint relations as possible into the active from, ie, into rules that can satisfy them. For those that cannot be converted into such rules, write rules that will test them for satisfaction.
- Write additional rules and modify existing ones to coordinate the actions of the rules (done in conjunction with the selected conflict resolution strategy).

In the following section, a canonical rule scheme for the synthesis problem is presented. See Ref. 30 for a detailed exposition of this methodology.

The constraints imposed by the design requirements can be classified into two basic categories: *convertable to active form,* ie, they can be converted into actions intended to satisfy them, and *passive.* The passive constraints do not guide or motivate any action. They do require satisfaction.

The synthesis problem is conceived as a search through the set of all pruned structures. These are candidates for the solution to the problem.

Assume that for each active constraint, a means of generating such candidates to test against the constraint is present. Call such an operator NEXT_IN_Ci.

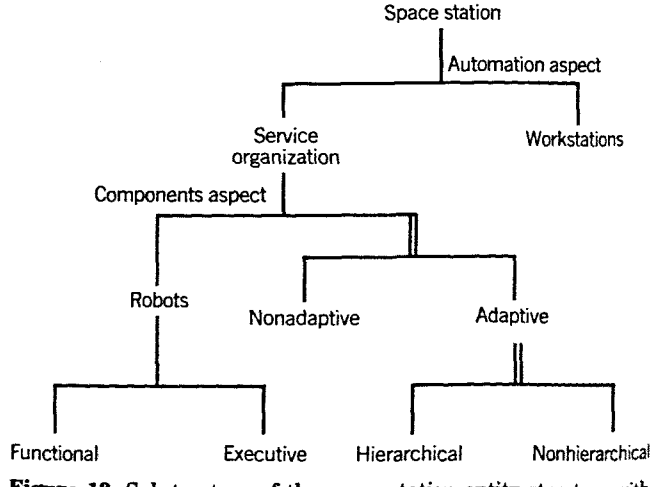


Figure 13. Substructure of the space station entity structure with specializations.

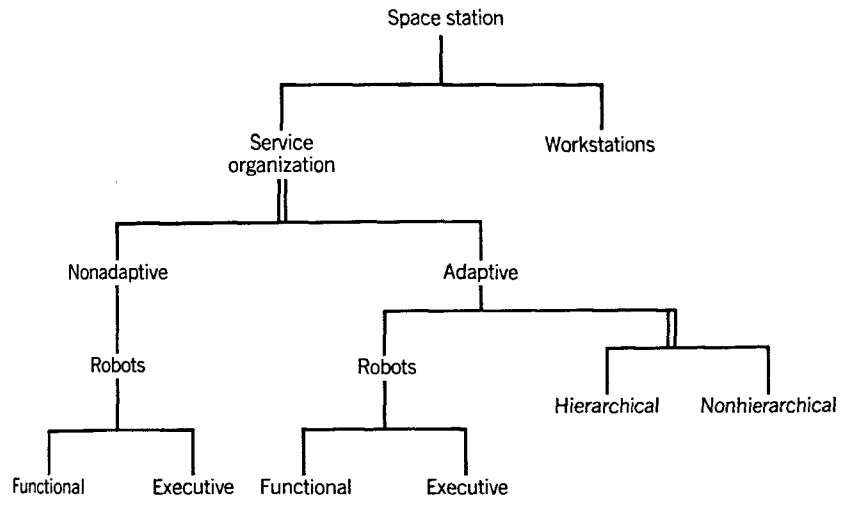


Figure 14. Moving down the aspect of service organization to its specialized entities.

The passive constraints have no corresponding operators and thus can only be tested for their satisfaction. Failure causes backtracking if a state has been reached for which none of the operators can be applied. Instead of applying an operator and then testing if it has consumed more than what remains of an available resource, the application of operators that would bring about the resource depletion is inhibited.

Con is a constraint that should be pretested. An operator, NEXT_IN_Ci, will map a state s into the region satisfying Con if, and only if, Con (NEXT_IN_Ci (s)). To allow the operator to be applied safely, it is necessary to define *applicability* predicate, Ai such that

Ai(s) if, and only if, Con(NEXT_IN_Ci (s))

Thus, a canonical rule scheme for a synthesis problem takes the following form:

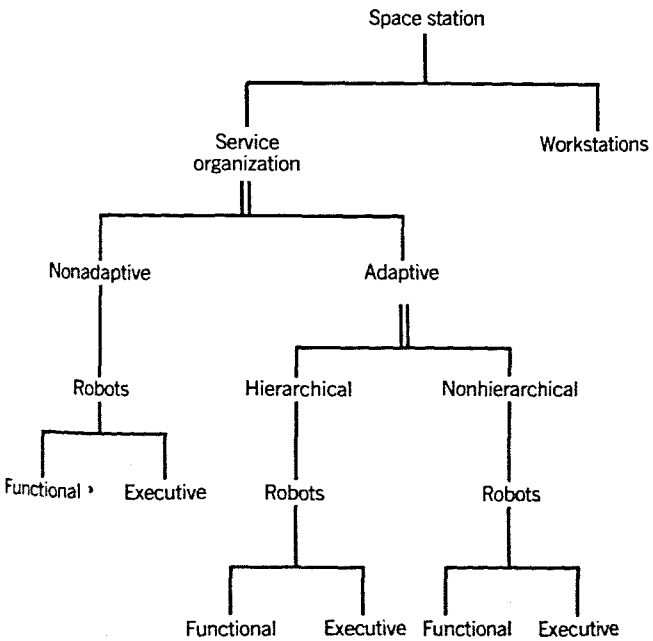


Figure 15. Design entity structure without specializations and aspects present at the same level down any path.

RC If C satisfied on (state)
then Output (state) as the solution
Rl If Cl is not satisfied
Al is satisfied
then state :=NEXT_IN_C1 (state)
.....
Ri If Ci is not satisfied
Ai is satisfied
then state :=NEXT_IN_Ci (state)
.....
Rn If Cn is not satisfied
An is satisfied
then state :=NEXT_IN_Cn (state)

To illustrate how the above scheme can be applied in model synthesis, use the space station example introduced in Formal Framework for Model-Based System Design. Recall that pruning the space station design entity structure of Figure 3 with respect to generic frame "Adaptability" results in the structures depicted in Figures 16a and 16b, respectively. Consider the synthesis of the station's model with regard to the automation aspect. For the design model structure of Figure 16, the following constraints are defined:

For the automation aspect of the entity station:

1. TOTAL.ROBOTS.OUTPUT >= TOTAL.STATION.WORK LOAD
2. COMPLEXITY.OF.ROBOT.ORGANIZATION <= MAXIMUM.COMPLEXITY

For the entity robots:

3. NUMBER.OF.ROBOTS IN [1,MAXIMUM].

The first constraint specifies that the robot organization should attain a level of productivity high enough to satisfy the station's work loads. The second constraint is imposed on the level of complexity of the organization. (Different measures of complexity are given in the frame Adaptability.) The third constraint says simply that the number of robots may not exceed a total number of robots available. This constraint can be further refined to apply to functional and executive robots. Similarly, more detailed constraints on the complexity

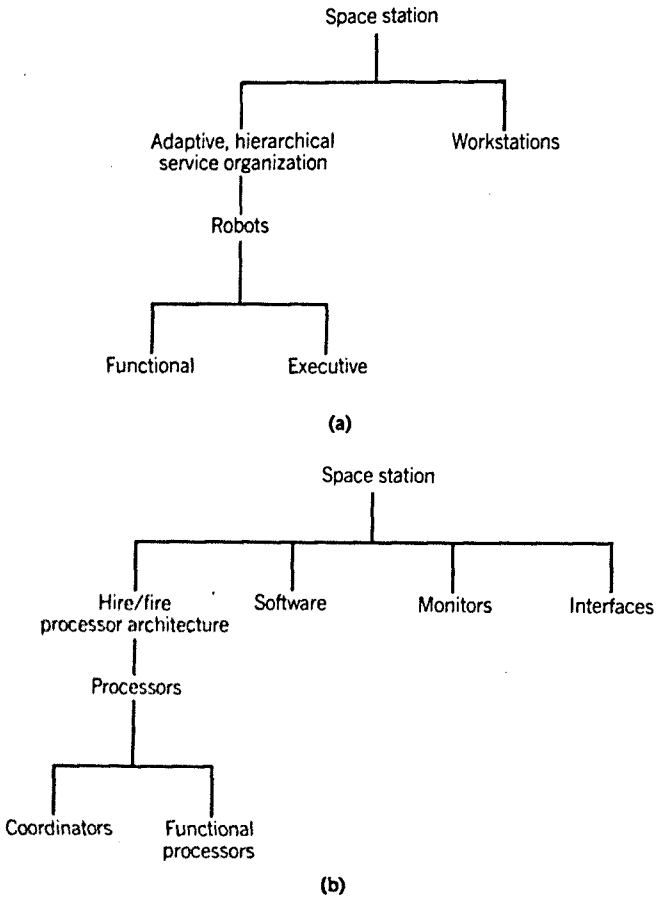


Figure 16. (a) Pruned model structure for space station design, automation aspect. (b) Pruned model structure, control aspect.

of the organization may be imposed once the reorganization schemes are known.

These constraints are converted into a production rule scheme according to the canonical form. Rule RC is the global constraint checker. Rules RC1 and RC2 are implemented as local constraint satisfiers.

```
RC if TOTAL.ROBOTS.OUTPUT >= TOTAL.STATION.WORK
      LOAD
      COMPLEXITY.OF.ROBOT.ORGANIZATION <= MAXI-
      MUM.COMPLEXITY

   then "STATION COMPLETED"

RC1 if ROBOT.AVAILABLE
      TOTAL.ROBOTS.OUTPUT < TOTAL.STATION.WORK
      LOAD

      then HIRE THIS ROBOT
           update COMPLEXITY.OF.ROBOT.ORGANIZATION

RC2 if REORGANIZATION IS FEASIBLE (with respect to
restructuring policy)
      COMPLEXITY.OF.ROBOT.ORGANIZATION >
      MAXIMUM.COMPLEXITY

   then
      REORGANIZE THE ROBOTS
      UPDATE TOTAL.ROBOTS.OUTPUT
```

After candidate structures that satisfy all the constraints have been found, design models of the station should be constructed and evaluated through simulation. The methodology for the model and experimental construction is discussed in detail in Refs. 7 and 30.

ENVIRONMENT FOR INTEGRATED, MODEL-BASED SYSTEM DESIGN

It has been our contention thoroughout the foregoing sections that the system entity structure and the concept of generic frame type constitute the knowledge that can support automatic construction of design models and experimental frames. To explain the argument, the following architecture for an expert system design environment is proposed. As illustrated in Figure 17, the data base of design objectives is one of the major components in the system. It must be well understood that the design objectives drive three fundamental processes in the methodology: first, the retrieval and/or construction of the design entity structure. (Naturally, the designer desires to obtain a family of design representations for a given set of objectives.) Secondly, the objectives serve as a basis for the specification of generic observation frames. Finally, the structural aspects of design generate a set of rules for the design model synthesis.

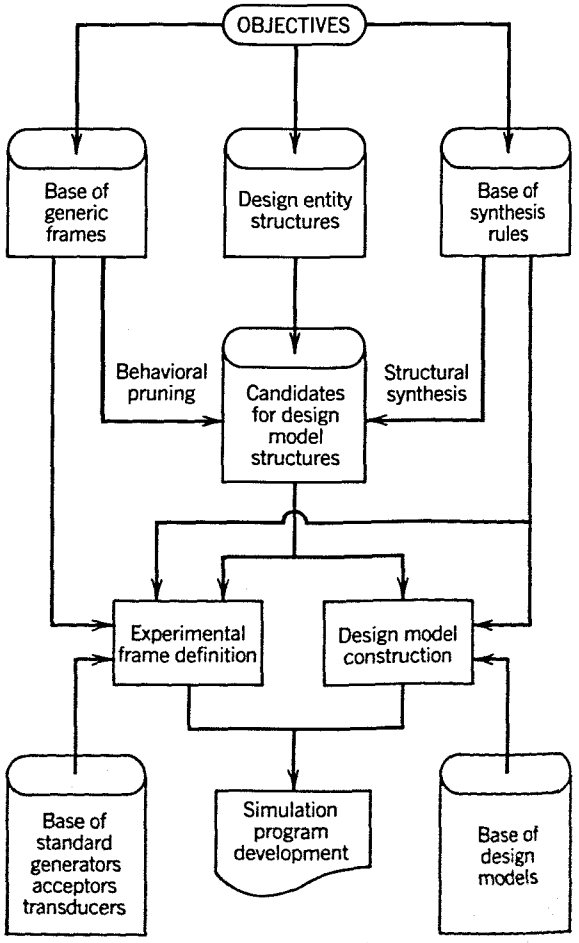


Figure 17. Integrated environment for system design support.

The ultimate purpose of the system depicted in Figure 17 is to analyze and integrate the relationships concerning the objectives specification base, the generic observation frame base, and the design entity structure. Such an integration should result in the formulation of design models and simulation experiments for a problem at hand.

The behavioral aspects of the design objectives are expressed in terms of generic observation frames. Pruning the design entity structure in corresponding observation frames results in substructures conforming to the behavioral objectives.

The substructures are then tested for satisfaction of synthesis rules that are derived from the design structural constraints as presented in Design Model Synthesis. Both behavioral and structural pruning applied to the design entity structure should result in design structures that are candidates for hierarchical model construction. The term candidates implies that some checks for consistency and admissibility (in the sense of conformance to the objectives) should be performed at this stage. If the candidate is inadmissible or no candidates can be obtained by pruning, the process should be reiterated with possible user intervention. The kinds of interventions suggested are modifications or retrieval of the new system entity structure, enhancement of the generic experimental frame, or modification of synthesis rules. The system should construct design models for the structures generated as a result of behavioral and structural synthesis employing the multifacetted model construction methodology presented earlier and in Ref. 1.

The design models should be evaluated through extensive simulation studies in experimental frames induced by the generic frame types. A detailed exposition concerning this aspect of design is presented in Ref. 30.

SUMMARY

In presenting the model-based system design methodology, the main focus is on two major aspects of the design process, the *design model development* and *specification of experimental circumstances* for design simulation.

Based on the formal concepts of the system entity structure and experimental frame, a framework for objectives-driven design model generation has been developed. In this framework, the behavioral aspects of design objectives are reflected in the *generic frame types,* which are prestructures for the experimental frames. The generic experimental frame types serve as a basic means of extracting model structures conforming to the behavioral objectives from the design entity structure. Effective *pruning procedures* have been developed to perform this task. The procedures have been further refined to extract model composition trees from the design entity structures in which specialization relations occur.

The structural aspects of the design objectives represented by a set of constraints have been shown to drive the process called *model synthesis* effectively. A canonical *production rule* scheme has been given for generating model synthesis rules.

These concepts are of a propositional nature. It is stressed that an attempt has been made to lay a foundation on which a design process can be based. A computer-aided expert design environment that internally represents the entity structures and generic frames and has means for dynamically manipulating these structures has been envisoned. Implementation of such a package, in all its generality, may be a long way off. However, efforts are under way to advance the design methodologies further.

BIBLIOGRAPHY

1. B. P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation,* Academic Press, Inc., New York, 1984.
2. M. S. Elzas, "The Use of Structured Design Methodology to Improve Realism in National Economic Planning," in H. Wedder, ed., *Model Adequacy,* Pergamon Press, Ltd., Oxford, UK, 1982.
3. T. I. Oren, "Computer Aided Modeling Systems," in F. E. Cellier, ed., *Progress in Modeling Simulation,* Academic Press, Inc., New York, 1982.
4. D. P. Siewiorek, D. Giuse, W. P. Birmingham, *Proposal for Research on DEMETER: A Design Methodology and Environment,* Carnegie-Mellon University, Pittsburgh, Pa., Jan. 1983.
5. J. W. Rozenblit and B. P. Zeigler, "Concepts for Knowledge-Based System Design Environments," *Proceedings of the 1985 Winter Simulation Conference,* San Francisco, Dec. 1985.
6. T. I. Ören, B. P. Zeigler, "Concepts for Advanced Simulation Systems," *Simulation,* 32(3), 69–82 (1979).
7. J. W. Rozenblit and B. P. Zeigler, "Entity-Based Structures for Model and Experimental Frame Construction," in M. S. Elzas and co-workers, eds., *Knowledge-Based Modeling and Simulation Methodologies,* North Holland Publishing Co., Amsterdam, 1986.
8. W. A. Wymore, *A Mathematical Theory of Systems Engineering—The Elements,* John Wiley & Sons, Inc., New York, 1967.
9. W. A. Wymore, *Systems Engineering Methodology for Interdisciplinary Teams,* John Wiley & Sons, Inc., New York, 1976.
10. M. Asimov, *Introduction to Design,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.
11. J. C. Enos and R. L. van Tilburg, "Software Design," in R. W. Jensen and C. C. Tonies, eds., *Software Engineering,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979.
12. G. Nadler, "An Assessment of Systems Methodology and Design," *Proceedings of The International Conference of The Society for General Systems Research,* Los Angeles, May 1985.
13. J. W. Rozenblit, "Experimental Frames for Distributed Simulation Architectures," *Proceedings of the 1985 SCS Multiconference,* San Diego, Jan. 1985.
14. M. Gonauser and A. Sauer, "Needs for High-Level Design Tools," *Proceedings of the 1983 IEEE Conference on Computer Design,* IEEE, New York.
15. J. W. Rozenblit, *Structures for a Model-Based System Design Environment, Technical Report,* Siemens AG, Munich, 1984 (internal distribution).
16. M. Gonauser, R. Kober, and W. Wenderoth, *A Methodology for Design of Digital Systems and Requirements for a Computer Aided System Design Environment, IFIP WG 10.0,* Sept. 1983.
17. R. Kober and W. Wenderoth, "Problems and Practical Experience in High-Level Design," in Ref. 14.
18. D. Belogus, "Multifacetted Modeling and Simulation: A Software Engineering Implementation," Doctoral Dissertation, Weizmann Institute of Science, Rehovot, Israel, 1985.
19. R. E. Shannon, R. Mayer, and H. H. Adelsberger, "Expert Systems and Simulation," *Simulation,* 44(6) (June 1985).
20. B. P. Zeigler, Y. V. Reddy, and T. I. Ören, *Knowledge Representation in Simulation Environments,* Academic Press, Inc., New York, in preparation.

21. B. P. Zeigler, "Knowledge Representation from Newton to Minsky and Beyond," *Applied Artificial Intelligence* **1**, 87–107, (1987).

22. B. P. Zeigler, D. Belogus, and A. Bolshoi, "ESP—An Interactive Tool for System Structuring," *Proceedings of the 1980 European Meeting on Cybernetics and Systems Research*, Hemisphere Press, Washington, D.C., 1980.

23. B. P. Zeigler and R. G. Reynolds, "Towards a Theory of Adaptive Computer Architectures," *Proceedings of the Distributed and Parallel Computation Conference*, Denver, May 1985.

24. S. A. Gregory, *The Design Method*, Butterworth Publishers, Ltd., London, 1966.

25. H. D. Hall, *A Methodology for Systems Engineering*, Van Nostrand Publishing Co., New York, 1972.

26. A. P. Sage, *Methodology for Large Scale Systems*, McGraw-Hill, Inc., New York, 1977.

27. W. A. Wymore, *A Mathematical Theory of Systems Design, Technical Report*, University of Arizona, Tucson, Ariz., 1980.

28. M. D. Mesarovic, D. Macko, and Y. Takahara, *Theory of Hierarchical, Multilevel System*, Academic Press, Inc., New York, 1970.

29. P. P. Fasang and M. Whelan, "A Perspective on the Levels of Methodologies in Digital System Design," in Ref. 14.

30. J. W. Rozenblit, "A Conceptual Basis for Model-Based System Design," Doctoral Dissertation, Wayne State University, Detroit, Mich., 1985.

General References

D. K. Baik, "Performance Evaluation of Hierarchical Simulators: Distributed Model Transformation and Mapping," Doctoral Dissertation, Dept. of Computer Science, Wayne State University, Detroit, Mich., 1986.

A. Concepcion, "Distributed Simulation on Multiprocessors: Specification, Design, and Architecture," Doctoral Dissertation, Dept. of Computer Science, Wayne State University, Detroit, Mich., 1985.

L. Dekker, "Concepts for An Advanced Parallel Simulation Architecture," in T. I. Ören, B. P. Zeigler, and M. S. Elzas, eds., *Simulation and Model-Based Methodologies: An Integrative View*, Springer-Verlag, New York, 1984.

J. R. Dixon, *Design Engineering: Inventness, Analysis and Decision Making*, McGraw-Hill, Inc., New York, 1966.

A. Javor, Proposals on the Structure of Simulation Systems," in A. Javor, ed., *Discrete Simulation and Related Fields*, North-Holland Publishing Co., Amsterdam 1982.

R. E. Kalman, P. L. Falb, and M. A. Arbib, *Topics in Mathematical Systems Theory*, McGraw-Hill, Inc., New York, 1969.

T. I. Ören, "GEST—A Modeling and Simulation Language Based on System Theoretic Concepts," in T. I. Ören, B. P. Zeigler, and M. S. Elzas, eds., *Simulation and Model-Based Methodologies: An Integrative View*, Springer-Verlag, New York, 1984.

R. Prather, *Discrete Mathematical Structures for Computer Science*, Houghton Mifflin Publishing Co., Boston, 1976.

Y. V. Reddy, M. S. Fox, and N. Husain, "Automating the Analysis of Simulations in KBS," in Ref. 13.

J. W. Rozenblit, "EXP—A Software Tool for Experimental Frame Specification in Discrete Event Modeling and Simulation," in Proceedings of the 1984 Summer Computer Simulation Conference, Boston, 1984, pp. 967–971.

R. H. Sprague and E. D. Carlson, *Buidling Effecting Decision Support Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1982.

P. H. Winston, *Artificial Intelligence*, Addison-Wesley Publishing Co., Inc., Reding, Mass., 1984.

B. P. Zeigler, "Structures for Model Based Simulation Systems," in T. I. Ören, B. P. Zeigler, and M. S. Elzas, eds., *Simulation and Model-Based Methodology: An Integrative View*, Springer-Verlag, New York, 1984.

DESIRABILITY OF ROBOTS

ANIL MITAL
University of Cincinnati
Cincinnati, Ohio

INTRODUCTION

The need to strive for higher productivity is perpetual. Reduction in production costs and greater competition in the international market are two main concerns behind the move of many manufacturers to automate their existing facilities. New technologies, such as FMS, CAD/CAM, and Robotics, are rapidly being implemented in medium and high volume manufacturing. Robots, which are the key supportive elements in automated factories and stand-alone manufacturing cells, are dominating such functions as welding, painting, and loading/unloading.

When first developed, robots were expected to replace workers only in hazardous environments. However, robots have also begun replacing workers in monotonous, highly repetitive, and unstimulating tasks. Existing statistics on robot growth indicate a tremendous increase in the U.S. robot population. With improved capabilities and lower costs, the market could expand to as many as 200,000 units per year (1). By 1990, the world robot population is expected to reach 1 million. The potential market for robots in the United States alone is anticipated to be about 400,000 units per year by that time (2).

Even though new technology has historically created more jobs and led to higher productivity, initial introduction of automation has caused, or is expected to cause, significant displacement (3). In the long run, however, the commercial use of robots is expected to follow this historical trend, although some evidence does raise doubt about a rosy future.

Several different sources forecast that wide-scale usage of robots will lead to worker displacement (see also HUMAN IMPACTS; EMPLOYMENT, IMPACT). According to one study (4), approximately 100,000 jobs may be lost in the U.S. auto industry alone in the 1980s. The Robot Institute of America of the Society of Manufacturing Engineers estimates that 440,000 workers will be displaced by the end of this century and that only 20,000 of these may expect to find another job through attrition or retraining (5). For example, only one fifth of employees laid off by the U.S. auto industries in 1979 has returned to work. A study supported by the Congressional Budget Office (6) predicted a displacement figure of 1 million by the early 1980s. The study conducted by the Ad Hoc Committee on Triple Revolution (7) predicts that the employment displacemet from automation would be so great that it would be necessary for the federal government to provide generous and costly income supports to a large fraction of the work force. Even though the last two studies may be politically motivated, they do not contradict the findings of the others mentioned. On