**Communication in Automated Engineering Design**

by

David V. Morse, Chris T. Hendrickson

EDRC 12-32-89
Carnegie Mellon University

# Communication in Automated Engineering Design

**David V. Morse**

**IBM Special Studies Program**
**Department of Civil Engineering**
**Carnegie Institute of Technology**
**Carnegie-Mellon University**
**Pittsburgh, Pennsylvania**

**Chris T. Hendrickson**

**Department of Civil Engineering**
**Carnegie Institute of Technology**
**Carnegie-Mellon University**
**Pittsburgh, Pennsylvania**

**June 9,1989**

# Table of Contents

## List of Figures

## 1. Introduction

Engineering design tasks represent a class of complex generative problems whose solutions depend upon the cooperative participation of multiple specialists. In a building design project, for example, many design professionals must contribute knowledge and expertise in order to ensure a successful end result. Typically, each specialist approaches a problem with reasonably complete understanding of his local design objectives and the methodology for attaining them, but the specialist has only superficial knowledge about the other domains with which he must communicate and interact. An individual expert may also possess an understanding of global design objectives; the methodology for attainment, however, cannot be defined *a priori,* since the "best" global solution usually represents a compromise among specialists whose preferred local solutions are not fully compatible. The form and substance of the compromise solution emerges from the interactive process itself.

The communications entailed in cooperative engineering design vary over a broad spectrum, ranging from straightforward dissemination of information to highly complex exchanges which characterize negotiation and compromise among contending design participants. Throughout this range of communication types, the goals remain constant: to ensure feasibility and efficiency in the global problem solution. Yet, the mechanism for achieving these goals varies as one traverses the spectrum. Simple information dissemination could be envisioned as a one-way communication; more commonly, two-way communication is appropriate to allow information recipients a channel for response. The response itself may vary from acquiescence to criticism to out-and-out rejection of the information received. In all but the first case, further action is required to ensure the desired solution consistency and efficiency.

The procedures required for consistency maintenance and conflict resolution at any particular level of complexity are largely dependent upon the degree of integrated design knowledge which is implicit in the problem-solving environment. If, for example, a cooperative design is to be performed by the combined efforts of disparate specialists having little knowledge of global goals and interdependences, then it can be expected that a referee will be required to resolve disputes even at the most elementary level. Conversely, a sophisticated group, possessing the capability to handle most consistency issues implicitly, would have less need for external arbitration. By extension, one could postulate a level of knowledge and sophistication where the need for external conflict resolution would be eliminated entirely, but such an ideal is seldom achievable in complex design environments. Thus, while the types of communications required in cooperative engineering design actually form a range in terms of complexity, they can be categorized into two subproblems, demarcated according to the intrinsic intelligence of a particular problem-solving system. We can speak of that subrange of consistency maintenance issues whose resolution is within the intrinsic capabilities of the system as *internally-enforceable* consistency issues and the subrange requiring external conflict resolution as *externally-enforceable* consistency issues.

In the interest of automating the engineering design process, considerable research effort has centered around the blackboard model for problem-solving, in which each contributing domain of expertise can be represented as a separate, semi-autonomous program which can contribute interactively to an evolving design solution represented in a shared global database (or blackboard). This model provides a reasonable abstraction of the traditional interdisciplinary engineering design process. One of the primary issues with the model, however, has been the maintenance of consistency in the emerging global solution, given that the contributions of each specialist can, and generally will, influence the contributions of others.

One consistency maintenance strategy that has been applied with some degree of success is the constraint-posting approach. In simple terms, this scheme gives temporal priority to the contributions of the various specialists: the information added to the global solution by one specialist becomes an input constraint on a specialist who contributes later. Clearly, this strategy runs the risk of eliminating some desirable solution alternatives from the feasible solution space: an inappropriate decision early in the design sequence can steer the ensuing design activity to a conclusion which, while consistent, might be unacceptable in practical terms. Modifications of the strict constraint-posting approach which provide for feedback loops or backtracking have addressed this concern to a certain extent.

This document describes another approach which views the issue in terms of the two subproblems described above, *internally-enforceable* and *externally-enforceable* consistency maintenance. Internally-enforceable consistency applies where the knowledge of design intent and interdependences represented in the problem-solving system is sufficient to permit automatic propagation of the consequences of an addition or modification to the emerging problem solution. Externally-enforceable consistency applies to situations where an addition or modification to an emerging problem solution creates a conflict whose resolution requires knowledge beyond the level represented internally; this would be analogous to the case in the conventional design office where a design dispute would require arbitration by a higher authority, due to the global nature or complexity of the decision factors. In this latter case, knowledge represented in the solution data structure would still be required to provide the arbiter with information necessary for rational decision-making.

The following section considers the communication issues which characterize cooperative interactive engineering design, in terms of the motivations as well as the mechanisms for information-sharing. Then, a conceptual overview is presented of a model which addresses the communication and representation requirements of complex design in the automated environment. The model consists of two components. The *process* component, or problem-solving framework, is a variation on the blackboard model which incorporates object-oriented programming. This component forms an architecture which facilitates communication and organizational requirements for cooperative, interactive, value-based engineering design. The *semantic* component provides a model for the abstraction of solution knowledge essential for consistency maintenance and value-based reasoning. Subsequent sections of this document describe a prototype which will illustrate the application of the model to the domain of floor layout and equipment placement.

## 2. Communication in the Design Process

When one considers the ingredients for a successful engineering design project, often the first need which comes to mind is the requirement for design expertise. This is a proper observation, since the design depends for its form and integrity upon the wealth of knowledge, experience and creativity of the contributing design experts. However, in complex designs characterized by the need for multiple domains of specialty, a second ingredient ranks equal in importance with the design expertise itself: the need for effective communication. Communication binds the contributions of individual experts into a cohesive whole. Many a project shortcoming, be it a matter of component inconsistency or failure in achieving global objectives, can not be attributed to a lack of design competence among individual contributors; rather, it is often traceable to the inadequacy of information exchange between them.

## 2.1. Uses of Communication

What **functions are served** by communication in cooperative engineering design? Most commonly, it is a means of disseminating information generated by an individual design discipline to others in the process who need it, either as input for their own design activities or in order to ensure consistency at design interfaces. Such exchange can promote design consistency by articulating constraints between related disciplines and by facilitating reactive adjustments in the evolving solution.

Communication is also essential in exploring hypothetical design alternatives, what is referred to in the vernacular as "what-if analysis." It is characteristic of design problems that numerous solution paths generally will lead to feasible results; yet, the various feasible solution alternatives can vary widely in terms of *how well* they meet overall design objectives. Given the strong motivation for design efficiency, the ability to consider alternate solution paths is useful. Communication plays a vital role in assessing the ramifications of hypothetical solution variations from the viewpoints of the various interdependent domain specialists.

Finally, communication is essential where the competing interests of individual design specialists threaten the consistency of the overall design solution. Conflict, in this sense, is a positive force, since it represents the attempts of individual designers to optimize with respect to local priorities; nonetheless, compromises among local priorities often are essential to maintain overall solution integrity. This requires conflict resolution, a special type of "what-if" analysis in which the contending parties submit to arbitration by a higher decision-making authority. The communication requirements of conflict resolution are highly complex, involving exchanges between arbiter and disputants to establish not only a feasible range of compromise solutions but also to determine the "best" alternative within that range. Frequently, the ramifications of the "what-iT analysis will extend significantly beyond the immediate parties to the conflict.

### 2.2. Modes of Communication

Communications enable the exchange of information between the multiple participants in the cooperative design process. Typical communication modes in the interactive design environment fall into five categories:

**Communication between a designer and the design record.** A design record refers to any media used to convey information about the emerging design solution. In the conventional engineering design environment, the design record consists of drawings, specifications, calculation sheets, models, etc. In the world of automated design, the record consists of one or more solution databases. The designer communicates with the record to represent his contributions to the solution and also to obtain required input information **for** his specific design activities.

**Communication between peer designers.** Though much of the information exchange between individual design specialties takes place indirectly via the design record, there also is a frequent need for direct communication between peer designers. One purpose of such an exchange might be to obtain clarification or to raise an objection to another expert's solution input, in the interest of overall design consistency. Another purpose would be to obtain information or input from another design specialist with regard to intent, in advance of its detailed design activity and formal entry of solution data into the design record. In the conventional engineering environment, this might take the form of visiting another design

department to take a look at their work while it is still "on the boards."

**Communication between the project manager and the design record.** In a typical complex design, the project manager will monitor the emerging solution in order to ensure consistency with global project objectives. This is considered a "read-only" access to the design record, since it typically is inadvisable for the manager to make design modifications without first consulting the appropriate specialists. The project manager can view and query the design record, possibly leading to communications of the fourth type:

**Communication between designers and project manager.** The relation between designer and project manager is usually one of interdependence: while the designer possesses requisite specific technical knowledge to ensure a feasible solution, the project manager maintains the global perspective and decision-making authority to direct the integrated solution in accordance with overall project priorities and objectives. As such, the communication between the two usually takes a form where the manager queries the specialist in order to obtain sufficient understanding of technical issues germane to a particular global decision-making situation. Such an exchange might be initiated by either party. The manager may raise questions as a result of reviewing the design record; individual designers may involve the project manager as arbiter to resolve conflicts among contending local design issues. In either case, the designers generally function as advisors and the project manager is responsible for the ultimate decision.

**Communication within the design record.** Various mechanisms have been employed to impose control and consistency on the design record itself. The use of design standards is one common example. Another method of promoting consistency in conventional building design is the traditional "pin-overlay" technique, where drawing transparencies from interfacing disciplines are physically superimposed to check for spatial conformity. The automated design environment, with its potential for incorporating knowledge on interdependencies directly into the solution database structure, offers a significant opportunity for sophisticated consistency maintence *within* the design record.

## 2.3. Semantics of Communication

Clearly, the required content of a given communication varies with the mode. Simple broadcast of a design result may require little more than physical attribute data, while an effective interchange for the purpose of conflict resolution would encompass information on design intent, rationale, propensity to modification, and many other factors. While each specific design domain would have distinct requirements at the detail level, **a** requirement for the following classes of information is common to all:

**Physical attributes** convey information used for direct processing or computation, such as dimensions, section properties, material characteristics, etc.

**Constraint Information** expresses the nature of relationships that must be observed and maintained to ensure a feasible design solution. Constraints may be of several forms:

- *Objectives,* which express the desirability or acceptability of a particular solution value relative to design goals and priorities,
- *interdependencies,* which express the legal relationships between various design entities, and

- *Limits,* which express the feasible bounds on a solution value (e.g., code restrictions or physical limitations on the design).

**Design firmness** expresses the level of design detail associated with a solution entity; this is helpful in conflict resolution decision-making, as it bears upon the reliability and flexibility of solution data. Information coming out of a preliminary design would normally be viewed differently from information representing detailed consideration and computational effort; detailed design information would be handled differently from information on a built design.

**Design sensitivity** information is essential to rational decision-making, given the reality that a globally-consistent design invariably requires compromise among the individual design participants. Sensitivity refers to the feasibility and cost associated with modification of a design solution. It has an explicit component: evaluation metrics can be applied to a particular solution entity and its variations, to assess its *direct* sensitivity to change. It also has an implicit component which arises from the interdependences within the design: modification of a component may have cost-relative ramifications beyond the directly-assessable effect.

The problem-solving model which is introduced in the following section addresses these communication characteristics and requirements within the context of an automated system for cooperative engineering design. It employs a semantic data representation scheme comprised of five elements:

- **Data Attributes.** This category contains information germane to direct computational and inferential processing of the data object itself: such information as the value of the data item, units, and its justification (i.e., which design operator placed the data item, whether the current value is a result of arbitration, etc.). Methods for obtaining "derived" data attributes from basic data values can also be represented.
- **Data Interdependences.** This category contains information on the genealogical relationship of the data item with its ancestors and descendants in a hierarchical data structure as well as other domain-specific interdependencies which might exist (e.g., beam connectivity information in a structural floor system).
- **Compatibility Conditions.** Contains information on value ranges and functional relationships which must be maintained in order to preserve consistency in the global solution. This category would also include any mechanisms employed to flag data inconsistencies requiring external conflict resolution.
- **Level of Design Completeness.** A classification to delineate the state of design refinement for that portion of the design to which the subject data item belongs. For example, a beam data item might be classified as a preliminary design, a final design or an existing (built) member. The distinction would have a bearing on the sensitivity of the data item to change; this in turn would influence the decision strategy if the data item were a party to arbitration.
- **Evaluation Metrics.** Contains information on the "value profile[1]* for a data item: its cost or benefit in terms of the evaluation criteria defining the global design goals. Typically, the value(s) associated with a current data item value would be represented, along with the function(s) for computing the value(s) from relevant attributes and global costing information. In the case of a steel beam, for example, the cost function would compute the value based on the current attribute values for end-coordinates, structural section and global unit cost data for structural steel. A data object representing a class of objects would maintain the aggregate value of objects below it in the hierarchy. This information is used to facilitate value-based conflict resolution by permitting rational comparison of multiple design alternatives.

## 3. Problem-Solving Model

The model that is proposed for interactive, value-based problem solving in multidisciplinary engineering design is the **GUIDE** (Globally-Unified Interactive Design and Evaluation) model. It is based on the concept that interactive design will entail the cooperative involvement of multiple specialist agents, each which represents a separate center of expertise necessary to the success of the overall solution. The contributions of the individual agents can be expected to be interconstraining to various degrees; that is, the design decisions made by a certain agent will, in general, impose limits which will impact the decisions of other agents. Further, the GUIDE model is developed with the idea that it is not sufficient to develop a consistent solution-- one in which constraints are satisfied. In order to be useful in practical engineering design situations, the solution must be developed with consideration for efficiency and effectiveness as well, according to a value system which reflects the global objectives of the design project.

The GUIDE model considers two important aspects of the cooperative design environment. The first of these is the *process* component: the problem-solving framework which addresses the mechanics, communication and control issues involved in interactive multidisciplinary design. The second aspect of the model is the *semantic* component. This addresses representation issues involved in providing a data structure that will support the maintenance of solution consistency and also facilitate the value-based reasoning necessary to arrive at an efficient solution.

This section discusses the primary considerations associated with each of these components in the generalized GUIDE model. A detailed description of a prototype application using the GUIDE model is presented in the subsequent sections, to illustrate implementation details in applying the concepts to an actual engineering design domain.

### 3.1. Process Model

The GUIDE process model, or problem-solving framework, is an object-oriented blackboard structure. This structure consists of several components:

- **DESIGN OPERATORS.** As in most systems adopting a blackboard problem-solving architecture, design operators in the GUIDE model can be loosely-coupled autonomous agents, each of which embodies a certain specialized domain expertise relevant to the overall design task. These operators may be individual knowledge-based programs, algorithms or even individual rules. They may represent agents that have been developed specifically for the global design task at hand or may be disparate elements, such as pre-existing systems or the results of other development efforts, whose capabilities can be brought to bear usefully on the task at hand. In these latter cases, it is likely that a utility would be required to provide semantic translation between the data representation of the integrated system and that recognized by each disparate element.

  A second type of operator is inherent to the GUIDE model: the conflict resolution agent. This is an extension beyond the traditional view of blackboard knowledge sources in that the conflict resolver has authority over the design operators, to impose or modify constraints

when such action is necessary to preserve consistency of the solution. The conflict resolver plays the role of arbiter when two or more design operators place inconsistent information into the global solution database as a result of interfacing requirements and differing local design goals and priorities. Presumably, each design operator is attempting to optimize its piece of the design; when their "preferred" solutions cannot be satisfied concurrently, the conflict resolver is charged with evaluating various compromise alternatives in light of the global project priorities and imposing the most efficient resolution choice on the global solution. In order to accomplish this, the general conflict-resolution procedure is conceived as having the following form:

- A conflict situation is recognized and flagged by the global solution database, characterized by the condition that the defined legal relationship between two data items is not satisfied and provisions do not exist within the global solution data representation to resolve the discrepancy automatically. The contending data values in the global solution, through their expressed functional dependency, define two endpoints in a range of resolution alternatives. Hard constraints on either of the data values could further restrict the feasible solution alternatives to a subset of this range.
- From the feasible range of compromise resolution alternatives, the conflict resolver establishes a discrete set of evaluation points. This could be accomplished by a simple segmentation of the overall resolution range or by a more sophisticated "hill-climbing[1]* technique.
- The conflict resolver creates a "hypothetical world** representing each candidate conflict resolution alternative. The data values corresponding to the resolution alternative are imposed as solution constraints, and the affected design operators regenerate those portions of the design impacted by the new constraints. The valuation metrics associated with the global solution data allow an evaluation of each hypothetical alternative in terms of global design priorities, (In the case of highly involved designs, regeneration could be accomplished using a simplified interdependency network, to arrive at an approximate evaluation of each hypothetical design alternative while reducing the computational burden on the system.)
- The conflict resolver selects the feasible resolution alternative which best satisfies the global design objectives and imposes the corresponding "compromise" data values on the global solution. Portions of the solution impacted by the new constraint values are regenerated by the appropriate design operators.

Clearly, the conflict-resolution process can become extremely resource-intensive. The use of a simplified data dependency network addresses this issue to some extent. Further simplification is possible, at the expense of generality, by incorporating into the conflict resolver domain-specific knowledge relevant to the common forms of conflict that might be anticipated in particular cooperative design problems. A conflict resolver with practical efficiency could be considered as having a "toolkit" of domain-specific strategies to handle the common and predictable conflict situations germane to a particular problem type as well as a "fallback" general resolution strategy to cover other conflict situations.

Design operators in the GUIDE model, then, are conceived as each being focused on a particular subproblem within the global design project, with limited knowledge of the other subproblems and specialists contributing to the overall solution. Where the system knowledge is sufficient to handle consistency maintenance issues automatically, such knowledge resides in the global solution data structure in an implementation of the interdependency and compatibilfty-expression provisions of the semantic data model. Where external conflict resolution is required, the conflict resolver can query individual design operators for their domain-specific inputs on solution flexibility and sensitivity, but the task of interpreting these inputs with respect to global project priorities and arriving at the appropriate arbitration decision belongs to the conflict resolver. This division of labor

promotes a high degree of autonomy, and thus modularity, for the design operator modules of the problem-solving system.

- **BLACKBOARD.** The GUIDE blackboard is a global database, accessible by all design operators, which contains information on the evolving solution. One essential category of global **data** is **the** representation of the current state of the design. The GUIDE model is intended **for use** in so-called "routine design[1]" situations, characterized by the fact that the general *form* **of the** solution may be predefined. In conventional building design, for example, one can describe the components and interrelationships that comprise a building even before the first decision is made about layout or sizing. This implies the potential to invest the database with a considerable amount of intentional knowledge about the artifact to be designed. The GUIDE model capitalizes on this potential through the incorporation of a knowledge-rich semantic data model (discussion follows) and through the use of a communication-rich, object-oriented, frame-based data structure. The result is an intelligent global solution database which can:

  - **Maintain Internal consistency through its own knowledge of data Interdependences.** The attachment of active values, or demons, to data attributes permits certain consistency enforcement actions to be triggered automatically when an attribute is modified. The object-oriented blackboard facility permits direct messaging between data objects, so that information on attribute modifications can be communicated quickly and efficiently to all interdependent solution data objects which could be impacted by the modification.

  - **Assist the conflict resolver In Identifying Instances requiring external arbitration.** In addition to managing internal consistency, the data structure can incorporate knowledge which allows it to recognize when the consistency maintenance requirements fall outside the scope of its own automatic resolution procedures. Such an instance could be characterized by a condition where the defined functional dependency between two data attributes is violated, but the provisions for internal resolution are absent. In such a case, the object-oriented nature of the blackboard can accommodate messages directly from the inconsistent data objects to the conflict resolver. These messages could simply alert the conflict resolver of the need for its services or could provide additional information as well, to assist the conflict resolver in identifying the nature of the inconsistency and identifying appropriate resolution strategies.

  - **Assist the conflict resolver In value-based arbitration.** The global solution database incorporates individual and cumulative information on the impacts of current data attribute values (which represent components of the solution) with respect to the global design priorities and objectives. The conflict resolver can query objects, or classes of objects, through direct messaging and obtain information relative to the "value" of solution components. Similar value-based queries are possible in the consideration of multiple hypothetical conflict resolution alternatives. This facility is the basis for rational decision-making on the part of the conflict resolver.

  Another category of solution data which may be represented on the blackboard addresses control of **the** solution process. This may be a combination of predefined control information and a dynamic record of solution progress. The GUIDE model will accommodate multi-level control strategies.

- **CONTROL ELEMENTS.** The GUIDE model provides two types of control elements. The *monitor* is primarily responsible for identifying instances in the intelligent global solution database where inconsistencies in the evolving solution require external arbitration by the conflict resolver. This can occur actively (from the monitor's viewpoint) by scanning the database for inconsistency flags or passively through the reception of alarm messages from inconsistent data objects.

  The *controller/scheduler* ensures an orderly progression toward problem solution by

regulating the actions of the various design operators. The controller/scheduler is given information on which design operators are eligible to make contributions and decides which should be authorized to do so (and in what order). Conceptually, the sophistication of the controller/scheduler could range from simple queue management to highly complex strategic solution direction.

Inherent throughout this discussion has been the notion that a high degree of communication flexibility can be attained by modeling the blackboard structure for cooperative problem-solving in the object-oriented paradigm. In particular, the capability exists to communicate directly between individual data objects, control elements, and design operators. This introduces the freedom to utilize formal control structures optionally when they can benefit the problem-solving process and bypass them when a simpler control strategy is more beneficial.

### 3.2. Semantic Model

The foregoing discussion of the intelligent global solution database in the GUIDE model already has alluded to two primary requirements in cooperative engineering design. To recapitulate, these are:

- To **maintain solution consistency.** When a change occurs to a data object, appropriate measures should be taken to ensure the integrity of the emerging solution. In some cases, the consequences of a change are logical, predictable and unambiguous, albeit that they may also be extensive. In such cases, data objects may be invested with "intentional" knowledge that, in conjunction with a communication-rich process model, will allow the consequences of the change to be propagated to the appropriate extent automatically. In other cases, a change to a data object may create an inconsistency situation whereby a number of resolution options are possible and the intervention of an external decision-maker is appropriate. Data objects may possess the knowledge to make this distinction and bring such situations to the attention of an external arbiter.
- **To facilitate solution optlmallty.** When consistency maintenance is to be provided by an external decision-maker through arbitration, it is essential to the rational design process that the decision be consistent with global design objectives (cost minimization, for example). It is not sufficient for the arbiter to know which data objects are in disagreement; given that an adjustment will be imposed upon the contending data items in order to restore solution consistency, the arbiter must also know how *sensitive to change* the data items are, to promote a decision that considers solution efficiency as well as solution consistency. Three issues arise when considering how to resolve a conflict between two contending design alternatives:

    - •Does the compromise solution under consideration represent a feasible design alternative? Can the contending data items legally assume the values that this compromise would dictate?
    - • Does the compromise solution give proper regard to the -firmness" of the data? Given that design is a process of successive refinement, it is sensible to assign greater decision-carrying weight to data which represents more highly-refined portions of the design.
    - • Does the compromise solution resolve the conflict in accordance with the global objectives of the design? If cost-minimization (or aesthetics, or functionality) is a global priority, then the selected compromise solution should attempt to optimize with respect

to this goal.

The GUIDE model seeks the attainment of these design objectives by defining a general semantic model for the solution data structure. The model classifies solution data knowledge into five categories and defines the intent of each. Specific data attributes appropriate for each category will depend upon the domain of routine design for which an application is developed, and the syntax will further depend upon the programming environment which is selected. Any instance of the representation scheme, however, will consist of the same essential semantic elements:

- Data Attributes,
- Data Interdependencies,
- Compatibility Conditions,
- Level of Design Completeness, and
- Evaluation Metrics.

Types of information germane to each semantic category are as described in Section 2.3.

The following sections illustrate the application of the GUIDE model to a specific design domain.

## 4. Prototype System Description

FLEX (Floor Layout EXpert) is a prototype implementation of the GUIDE (Globally-Unified Interactive Design and Evaluation) problem-solving model described in the previous section. FLEX addresses the domain of equipment arrangement and brings together several elements of consideration in floor layout for industrial facilities: the configuration of structural floor framing, the placement of mechanical equipment, the sizing and adequacy of structural components, and the constructability of the overall system.

These elements, while closely interrelated, are seldom adequately coordinated in conventional design practice. Most commonly, structural framing arrangements are dictated by the requirements of a pre-established equipment layout with the somewhat parochial rationale that the purpose of a structural system is "to serve" the needs of the facility arrangement. Formal evaluation of the system's constructability often occurs far downstream in the design sequence, if it happens at all. Reconfiguration of the floor arrangement based on construction considerations, when evaluation occurs so late in the cycle, is typically impractical in all but the most dire circumstances. Clearly, the prospect of integrating these facets of the floor design process to allow timely feedback and evaluation offers significant potential advantages from the viewpoint of global project optimization.

The concept of optimization as proposed by the FLEX prototype takes a somewhat different definition than that recognized by the operations research community, in which optimization often assumes the form of a rigorous mathematical formulation. While it might be argued that the application of optimization theory

to a simple floor framing system could be represented practically by a formal mathematical technique, the class of interactive design problems which FLEX is built to represent cannot, in general, employ such precise approaches. When interdependencies become more complex, optimization takes the form distinguished in operations research theory by the concept of *approximate optimization:* that is, the evaluation of a discrete number of solution options within the feasible solution range and the choice of that option which best suits the decision criteria.

In addition to the far-reaching interdependencies which characterize multidisciplinary design problems, the design objectives themselves are often complex and may entail entire lists of priorities such as cost, aesthetic appeal and functionality. FLEX employs cost as its primary metric for value-based decision-making, a choice that typifies a broad range of industrial design projects.

The overall architecture of the FLEX prototype conforms with the GUIDE model for cooperative problem-solving, which was described in the previous section. The FLEX implementation of the model is depicted in Figure 4-1 and consists of six conceptual components: a global solution database, a collection of design operators, a semantic interpreter, a conflict resolver, a monitor and a controller/scheduler. Consistent with the purpose of a prototype, each of these components is implemented to an extent which captures the essential features and demonstrates the utility of the model in complex, cooperative problem-solving. The following discussion describes each of these components, both in terms of intent and in terms of its implementation details within the prototype system.

## 4.1. Global Solution Database

The global solution database, or blackboard, in FLEX represents the evolving solution to the floor arrangement and design problem in the form of a hierarchical structure of data objects. This data repository is accessed by the design operators both to obtain the necessary input data for the individual processes and also to record their output. The blackboard in FLEX may be considered an *intelligent* database in that the global solution data structure contains information on the domain-specific legal restrictions and consistency relationships that must be maintained between individual components of the solution (intentional knowledge). This permits the blackboard to promote solution integrity by taking an active role in consistency maintenance on two levels:

**Internally-enforceable consistency,** as described earlier, refers to the activity of making all the updates to the data structure which follow naturally and unambiguously from the modification or addition of a data item by a design operator. If the consequences of a change in a data value are known (i.e., the interdependencies and constraints are defined), and if there are no conflicting viewpoints about the consequences (i.e., the consequences are a matter of logic rather than of opinion), then the intelligent database should be able to handle the details of maintaining consistency without reliance on outside help. An example of such a situation arises when a load is added to a beam in the floor system. The immediate consequence is that the load is added to that beam's load list. The logical implications, however, extend much further. The beam's end-reactions, shear and bending moment will likely change. The required structural section may also change. In addition, these changes will propogate to any other beams into which the affected beam frames. The consequences are logical, predictable and unambiguous; the intelligent global database, infused with sufficient intentional knowledge, handles them automatically.

**Externally-enforceable consistency,** by contrast, applies to situations where the conflict resolution

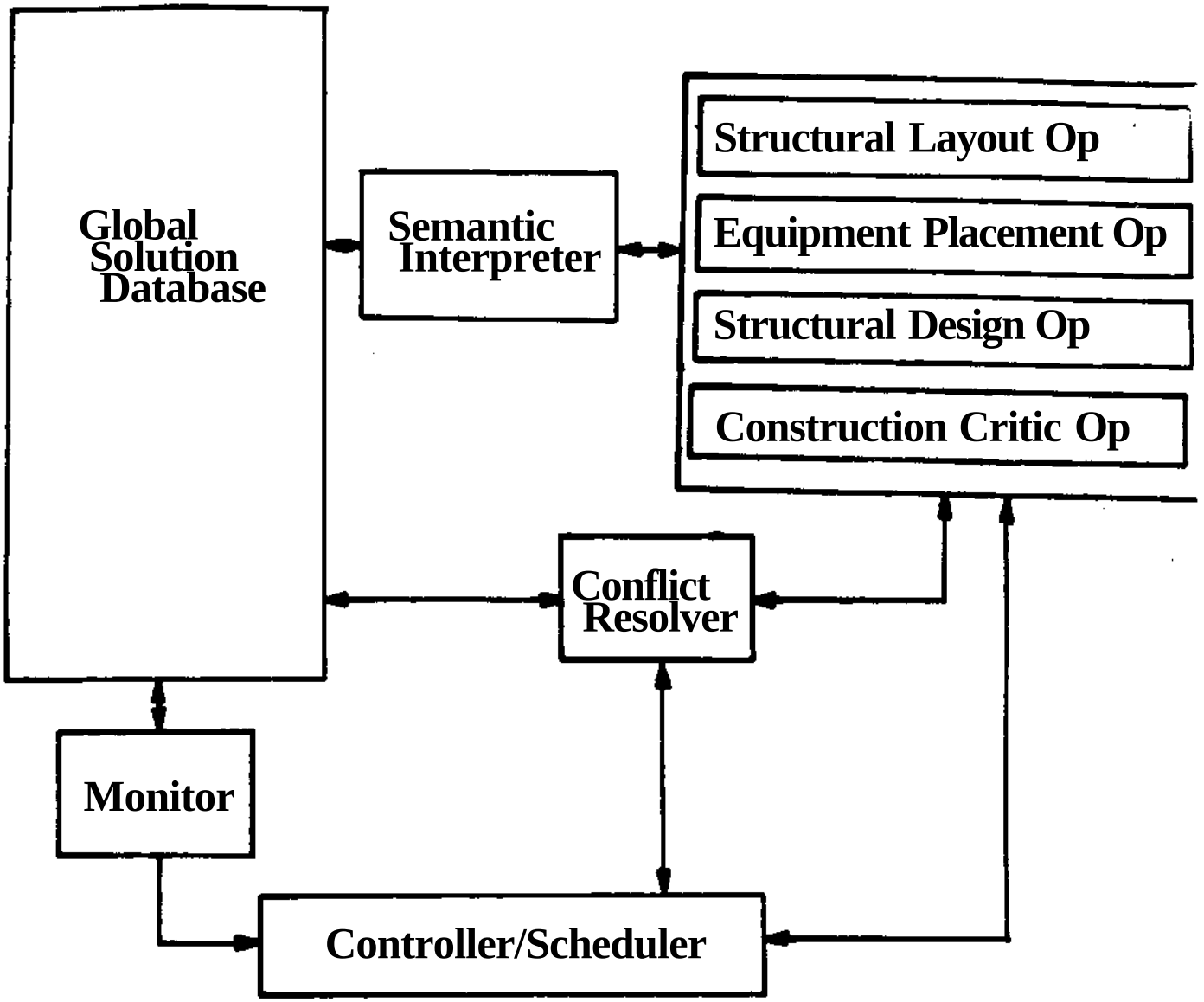



**Figure 4-1**: FLEX Prototype Architecture

decision requires not only knowledge of data interdependences but insight into overall project priorities as well: a value judgement, or tradeoff, is necessary. Given that priorities change from one project to the next, this value-based conflict-resolution knowledge should be modeled separately from the invariant intentional data knowledge. FLEX embodies this information in the *conflict resolver* module which will be discussed presently. For an example of a typical floor-layout scenario where external conflict resolution is predicated, consider the case of an equipment placement expert who wishes to support his equipment at a point where a beam does not exist in the structural system. In order to maintain design consistency, a resolution strategy must be selected from among several possible courses of action:

- The equipment placement expert could be directed to relocate the equipment such that it can be supported by the existing beam arrangement. Presumably this would represent a suboptimal, and therefore more expensive, alternative for the equipment expert (otherwise, he would have chosen the alternate location in the first place).
- The structural framing expert could be instructed to modify the framing arrangement so as to provide support beams for the equipment as currently located. This alternative would presumably entail additional expense for the floor framing system, since it would represent a move away from the structural framer's preferred solution.
- A compromise could be identified, whereby each party to the conflict would be required to make modifications in whatever combination would have the least adverse impact on overall project cost.

In a situation involving multiple feasible alternatives, the global solution database relies upon an external agent to make the requisite value judgements. The database, however, must possess the intelligence to recognize when external conflict resolution is appropriate and necessary, and it must also model the value-related attributes of its data such that an external conflict resolver can make decisions in a rational manner.

The structure of the global solution database for the FLEX prototype system is depicted in Figure 4-2. The relationship between individual data items reflects the inherent hierarchies in the major components of the floor layout domain: a floor system is comprised of structural components, supports and externally-applied loads. In this case, all of the structural components are beams, support points are represented by columns, and external loads are applied by pieces of mechanical equipment which must be supported. Each individual data item has a number of attributes which are represented using a frame-based approach, such that individual attributes can be attached to specific data items while inheritance permits an economical and consistent representation of those attributes which are common to whole classes of data items. These attributes represent the five classes of data description identified in the semantic component of the GUIDE model:

- Fundamental Data Attributes (units, magnitude, etc.);
- Relationships with Other Data Items (logical interdependencies);
- Compatability Conditions (legal relationships);
- Level of Design Completeness (preliminary, final, existing, etc.); and
- Evaluation Metrics (such as cost).

In addition, any data item attribute can have several pieces of information associated with it, represented as subslots or attribute *facets.* Figure 4-3 illustrates the attribute information associated with

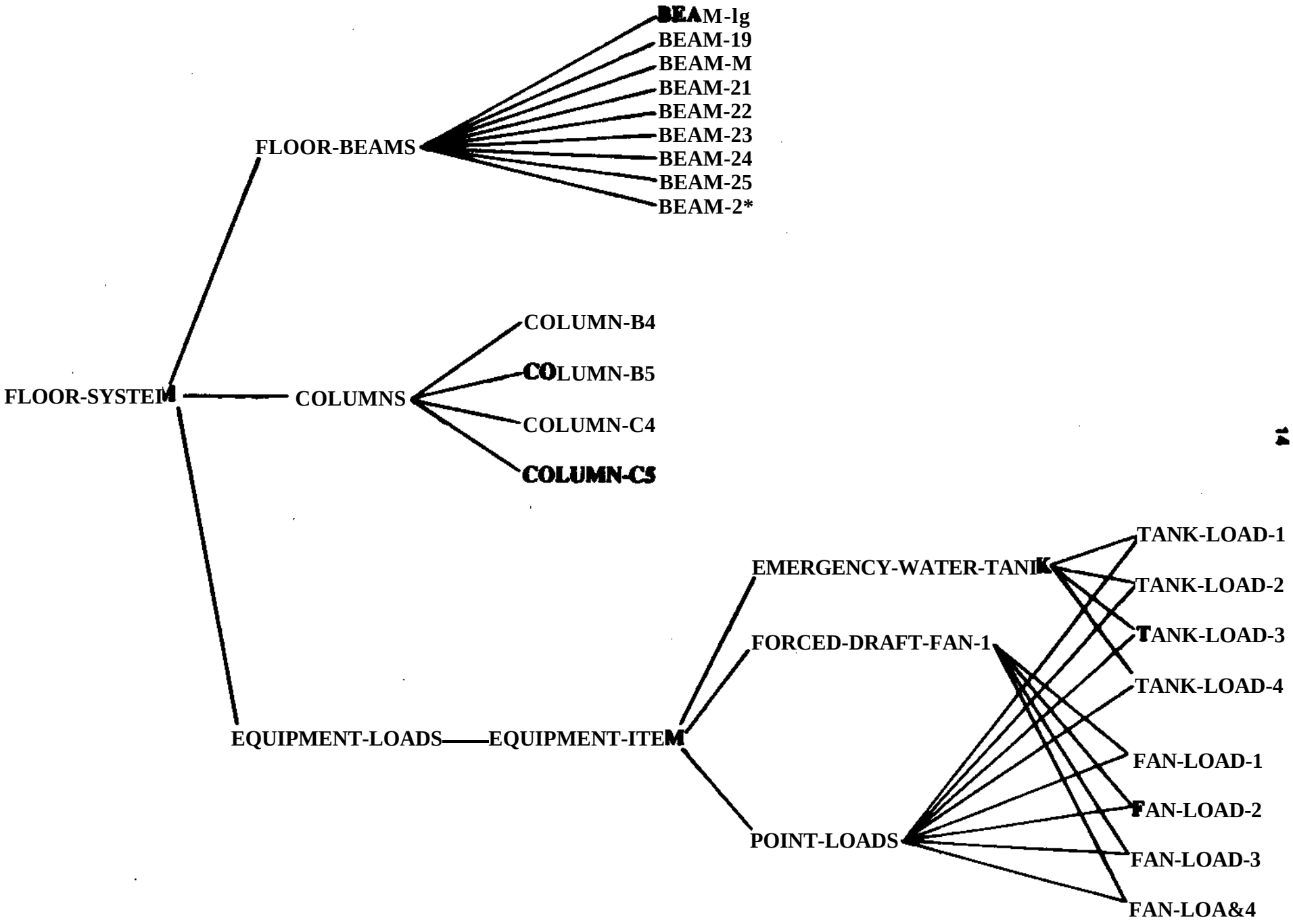



**Figure 4-2:** Example ot FLEX Global Solution Data Structure

a typical beam data item, and Figure 4-4 provides an example of the facets which might be associated with a particular attribute.

Consistency maintenance is facilitated through a communication strategy which includes a combination of active value attributes and object-oriented programming. *Active values* (sometimes called *demons)* cause actions to be taken whenever a change in value occurs to the data attribute with which the active value is associated. For example, suppose the length of a beam changes (signified by a change in its end-coordinates). The beam itself should be made aware that such a change in length may have such consequences as a change in reactions, shear, moment and section requirements. In addition, any "parent beams"* into which the affected beam frames will experience similar impacts, and so will *their* "parent beams", etc., until the ripple effect finally reaches the columns. Additionally, each data item possesses *methods,* or attached procedures, which enable it to make the pertinent self-evaluations. For example, if a beam becomes aware that its loads have changed, it has a method it can use to recalculate its bending moment.

The final piece in the story is the communication link. We now know, for example, that a beam can become aware when its loads have changed (through the use of active values), and the beam *knows* which other data items are impacted (through the knowledge defined by the semantic model). We also know that the other data items can *respond* to change by employing their *methods.* What remains is to provide a means for the beam to make the other impacted data items *aware* that a change has occurred. This is accomplished by the direct communication that can take place between data items by *messaging* in the object-oriented programming paradigm. In our example, the modeling of individual data items as *objects* allows the beam whose loads have changed to send a message directly to its "parent beams" which, in essence, would say: "My loads just changed. You support me. Therefore, you had best re-evaluate your bending moment as a result of this change."* Similarly, in an instance where the services of an external conflict resolver are required, a message can be employed to invoke the conflict resolution module.

## 4.2. Design Operators

The design knowledge for the FLEX prototype system is supplied by four design operators, each representing a specialized body of expertise pertinent to floor layout and design. Two of these, the structural layout operator and the equipment placement operator, could be considered as "facility layout"* experts, in that their primary concern is an acceptable and consistent spatial arrangement of floor beams and mechanical equipment. Spatial consistency, in this case, is predicated upon the requirements that sufficient clear space exists between any two pieces of equipment and floor beams are provided beneath equipment support points. The acceptability criterion for a design has these spatial consistency requirements as a prerequisite and, additionally, demands that the design be cost-effective. It is in the assessment of cost-effectiveness that two other experts come into play: the structural design operator, which designs the beam sections, thereby allowing a cost estimate of the structural system, and the construction critic operator, which evaluates the constructability criteria for the overall floor system, thereby providing timely feedback on the merits of the design from the perspective of construction cost.

Each of the design operators has direct access to the global solution database, with interpretive assistance as required (the semantic interpreter is described in the following section). A monitor module

**BEAM-20**

*beam-id*
*beam-section*
*begin-coord-x*
*begin-coord-y*
*end-coord-x*
*end-coord-y*
*begin-reaction*
*end-reaction*
*uniform-load*

*begin-support-member*
*end-support-member*
*beam-begin-children*
*beam-end-children*
*equipment-point-loads*

*methods:* *add-child-beam*
*remove-child-beam*
*add-equipment-load*
*remove-equipment-load*
*compute-cost*
*compute-length*
*compute-max-shear*
*compute-max-moment*

*design-level*

*beam-cost*
*floor-beam-unit-cost*

**Figure 4-3:** Example of Data Item Attributes in FLEX

**BEGIN-COORD-X from BEAM-20**

***Comment:*** **x-coordinate defining beginning of beam.**

*Values:* 540

***Units:*** **INCHES**

*ValueClass:* NUMBER

*Cardinality.Min:* 1

*CardinalityMax:* **1**

*Justification:* **FRAMER**

***Inheritance:*** **OVERRIDE.VALUES**

**Atwite; BEAM-COST-AV**
**REPLACE-VARIABLE-VALUE-AV**
**ADJUST-BEGIN-CHELDREN-AV**
**ADJUST-END-CHILDREN-AV**

**Flgurt 4-4:** Example of Attribute Facets in FLEX

(also described subsequently) determines when a design operator should be activated, based on the current state of the emerging global solution. The following paragraphs describe each of the design operators, and their implementation in the FLEX prototype, in further detail.

**STRUCTURAL LAYOUT OPERATOR.** Taking as its initial design constraint the support (column) locations, the structural layout expert attempts to configure a beam system to support the uniform load of the floor and any prespecified concentrated loads. The local design objective is to minimize the total weight of the structural system. The structural layout operator places information in the global solution database on beam end coordinates, connectivity, initial loading and design level (i.e., whether this represents preliminary design data, detailed design data, or data on an existing structural element). The structural layout operator also supplies the unit cost of structural steel, used for subsequent cost estimates. In interaction with the other design operators, the structural layout operator can respond by making local adjustments to its layout or by initiating complete new designs with additional constraints, as the situation dictates. For example, if all but one equipment load were to be supported by an existing floor layout, the logical response would be to add or move a single beam to meet the requirement. If sweeping changes were required, however, the more logical approach might be to redesign "from scratch" with the new requirements incorporated at the start. In the FLEX prototype, structural layout expertise is represented by a human expert interacting with the global solution database.

**EQUIPMENT PLACEMENT OPERATOR.** The design operator responsible for locating mechanical equipment on the floor takes as its input a list of equipment and associated placement constraints and parameters. The primary constraint on an individual equipment item addresses the concern of overall spatial consistency; each piece of equipment has an associated "clearance envelope" that defines a minimum distance which must be maintained from columns and other equipment items. The location parameters for each equipment item include a preferred location and a cost function which describes the dollar penalty associated with variance from the preferred location. This information, stored in the global solution database, permits the equipment placement operator to arrive at a consistent, "preferred" arrangement. When an item's location is established, this information is added to the global solution database, along with the associated cost of the selected location. Information on loads and locations of equipment support points is also entered (the magnitude and relative spacing of support points is predefined, so their absolute locations can be established once an equipment reference location has been determined).

[NOTE: The first iteration in equipment location, then, considers the optimum functional mechanical arrangement without regard to the structural layout. Only sheer coincidence would result in global design consistency on the first pass. More likely, adjustments will be required in the structural layout, the equipment layout, or both. The solution database incorporates the information required to assess the economic impacts of such adjustments.]

**STRUCTURAL DESIGN OPERATOR.** When a structural framing layout is first established, or when subsequent modifications occur to an existing layout due to changes in spatial arrangement or imposed loadings on members, it is necessary to consider the effects of these actions on the design of the impacted members. The intentional knowledge structure of the global solution database is sufficient to ensure that the effects of a change are propagated correctly; the beam data items themselves have the method-encoded knowledge to compute their design parameters, such as maximum shear, maximum moment and unbraced length. The structural design operator, then, is responsible for taking these inputs

when a beam has been cited for redesign and selecting the best structural section in accordance with governing design specifications. For the purpose of member design in the FLEX prototype, a knowledge-based system will be employed to select appropriate steel beam sections from a database of available rolled shapes in accordance with the strength and serviceability requirements of the American Institute of Steel Construction (AISC) specifications.

**CONSTRUCTION CRITIC OPERATOR.** The idea of assessing constructability issues during the layout phase of floor design recognizes that there is an additional cost component which is often overlooked in the early stages of a project- and which sometimes haunts the project later. While the structural layout and equipment placement operators both make cost evaluations, these are mainly from the material procurement and fabrication viewpoint. The construction critic, by contrast, considers design cost from the installation point of view. For the FLEX system prototype, construction criticism focuses on the complexity of the design (e.g., number of beam end-connections to be made) and produces output consisting of comments and suggestions for possible improvement, directed primarily to the structural layout expert. In the sense that the FLEX prototype is offered as an example of design integration, the construction critic operator exemplifies a whole conceptual class of "peripheral" design considerations whose closer linkage would benefit global project objectives.

### 4.3. **Semantic Interpreter**

The semantic interpreter module in FLEX is conceptualized as a simple utility to make syntactic translations between the representation of information in the global solution database and the semantlealIy-equivalent form used by any design operator whose representation scheme is dissimilar. A logical expansion of the FLEX prototype, for example, might benefit from the inclusion of additional design operators developed outside the system environment and therefore not "native" to FLEX. It would be expected that such disparate operators might express logically equivalent information in dissimilar forms. The semantic interpreter provides the two-way translation. In those cases where design operators are built using the native representation, of course, interpretation is unnecessary; the provision of a semantic interpretation module in the system architecture recognizes the conceptual requirement for true system modularity.

### 4.4. Conflict Resolver

The conflict resolver may be considered as a separate operator with a higher level of authority in the problem-solving process than the individual design operators. It is the arbiter with knowledge of global project priorities and the power to impose consistency in externally-enforceable conflict situations accordingly. The conflict resolver is inactive until the intelligent global database indicates a consistency-maintenance situation requiring extemal resolution (denoted by *status: inconsistent* flags on data items in the solution database). The flagged data items typically will identify a situation where proposed local solution alternatives cannot be satisfied concurrently.

In the general case, it would be reasonable to assume that the contending data items demarcate limits on a space of possible consistent "compromise" solutions. If this segment of the solution space is continuous, then an infinite number of consistent solution alternatives exists, and it is the task of the conflict resolver to select discrete points along the continuum as candidate solutions and to evaluate

relative suitability of the candidates with respect to global project priorities. A practical example would involve the bay spacing in a floor system, where facility layout preferences might advocate a very large space between columns (to minimize obstructions) and the structural engineer might opt for a closer column spacing (to achieve economy in the floor design). Any dimension for bay spacing between the two contending extremes would be a possible solution. In the absence of an exact mathematical optimization function (which is rare in cases of practical complexity), the strategy that will make a value-based design decision tractable will involve the evaluation of the total project cost at each extreme and at several intermediate discrete bay spacing values, followed by the selection of the "best" alternative, according to whatever global decision criteria are appropriate.

Of course, not all solution spaces are characterized by a continuum of possible and practical solution alternatives. It behooves a conflict resolver to have some specific knowledge of its domain in order to identify typical situations which would be better addressed by other conflict-resolution strategies. The FLEX prototype illustrates one such situation in the floor layout domain, where an equipment support might be located at a point where a beam does not exist to support it. If one were to follow the general (default) conflict-resolution strategy, such a circumstance would pose an infinite number of possible resolution alternatives:

- Move a beam to the current equipment support location,
- Move the equipment load to a current beam location,
- Add a new beam at the current equipment support location,
- Split the difference: move a beam and the equipment point load in various combinations until they coincide.

Each of these resolution alternatives represents a path to a consistent solution at a definable cost. In practicality, however, a competent designer would limit his strategies to the first three alternatives, recognizing that the myriad possibilities generated by the fourth alternative add great complexity to the decision process, with questionable potential return. Investing a conflict resolver with similar domain-specific knowledge to cover the more common and anticipated conflict types is a departure from generality that can be more than offset by the consequent gains in efficiency and rationality.

Thus the conflict resolver as conceptualized for the FLEX prototype has a general conflict-resolution strategy that entails the discretization and periodic evaluation of a continuous range of solution alternatives; in addition, however, it has the capability to recognize common conflict situations specific to the floor layout domain and to invoke specialized resolution strategies where appropriate. For the FLEX prototype, the conflict resolver will be simulated through user interaction with the database and design operators. An approach is being explored whereby each individual alternative in a conflict resolution session would be represented in a separate hypothetical "world" which would be maintained only until the best alternative could be identified and the global solution database modified to suit.

Several issues in conflict resolution, broached by the foregoing description, are worthy of further comment. First, the notion of selecting discrete candidate solution points for evaluation from a continuous range of possible solutions carries the implication that the best alternative will only approximate the true optimal value. Logically; the approximation could be improved by increasing the number of discrete evaluation points, but such a decision has system performance impacts. One possible middle ground

would entail the use of nonlinear search strategies within the solution space, reducing the search increment as "promising" regions of the feasible solution range are identified.

Second, for practical situations, design cannot be considered as a "single-pass" operation. Some portions of the solution database may represent the results of detailed design while others reflect only a preliminary treatment. At the extreme, in the floor layout application, some solution data may represent floor components which have already been built. The conflict resolver should recognize the level of design completeness associated with contending data items as a part of its resolution strategy.

Third, it is worthwhile to note in the global solution database when a data item value has been imposed as a result of arbitration by the conflict resolver. While this status would not exempt the data item from further adjustment, it would be useful in tracking the justification for the evolving solution.

Finally, the generation of multiple complete designs in order to weigh the relative attractiveness of various resolution alternatives is an approach that could easily become resource-constrained in large, complex design problems. One means of "pruning" the decision tree for evaluation purposes would be to differentiate between data interdependencies which are significant in the evaluation process and those which need not be considered except for consistency in the final design. This two-layer representation of the design interdependency network would add a nominal load to the data storage memory requirements at a significant potential savings in processing requirements for rational conflict resolution.

### 4.5. Monitor

In the FLEX prototype, the monitor functions to determine when it is appropriate for a design operator to be activated. The decision is based upon two factors:

- When sufficient information exists in the global solution database to satisfy the preconditions and inputs for a design operator to be invoked, or
- When the conditions of the evolving solution *require* a qualified design operator to be invoked.

As an example of the former situation, the equipment placement operator might be determined as eligible to fire when the following conditions are satisfied:

```
IF [there is physical data on an item of equipment <x>]
AND [there is not location data on equipment <x>]

THEN [equipment-placement-operator is eligible to run]
```

The latter situation refers to certain instances where the knowledge in the intelligent global database enables a direct determination that a design operator should be activated. For example, when inconsistencies arise between equipment support locations and beam locations, it is necessary to invoke the conflict resolver. It is desirable for this to happen expeditiously, regardless of the eligibility of other design operators to fire. The monitor should be able to recognize such a situation and instruct the controller/scheduler accordingly.

In some cases, the flexibility of the object-oriented environment allows the global solution database to

perform the monitoring process directly. For example, when the conditions of a beam data object change such that redesign is dictated, the data object can send a message directly to the structural design operator to obtain **the** desired **new** design information. Thus, the formality of involving the monitor and controller/scheduler **can** be included where it benefits the solution process and sidestepped where not needed.

### 4.6. Controller/Scheduler

The intent of the controller/scheduler module in FLEX is to regulate the interactions and contributions of the individual components in the model in order to ensure an orderly and consistent progression toward the complete global solution. Conceptually, the GUIDE model admits a considerable amount of sophistication to the controller/scheduler in terms of developing and pursuing solution directions and control strategies. The FLEX prototype implementation of a controller/scheduler functions primarily as an agenda controller acting upon design operator eligibility information provided by the monitor. If more than one operator is eligible to run at any given time, the controller/scheduler creates an agenda which, in the case of the FLEX prototype, consists of a queue. In the situation where a flagged inconsistency requiring external conflict resolution is identified in the global solution database, the controller/scheduler invokes the conflict resolver module and suspends further firing of design operators until the inconsistency has been resolved.

For the FLEX prototype, the controller/scheduler willl be simulated by user interaction with **the** individual design operators and the conceptual conflict resolver module.

## 5. Example Problem

A small sample problem will be implemented using the FLEX prototype system, to illustrate the features and component interactions described in the previous section.

### 5.1. Problem Description

A 15-by-20-foot bay of a proposed industrial facility will be considered, in which two items of equipment, a forced-draft fan and an emergency water tank, are to be placed. Neither the floor system configuration nor the equipment location plan is predefined, although "preferred[1]* location knowledge is incorporated in **the** data model **for** each piece of equipment. Such information reflects the fact that the equipment does not stand in isolation but must be connected to other equipment items by piping, ductwork, etc., **and** "preferred" locations represent the motivation to minimize connection costs. Each piece of equipment also has an associated cost function which reflects the additional connection expenses incurred in varying from the "preferred** location. (Optimization theory terms this a *penalty function.)*

Each design operator has a local objective which is its driving force in advocating a specific solution direction. The structural layout operator attempts to develop a beam framing configuration which minimizes the total weight of structural steel (a simplistic but reasonably valid index to the cost of the structural system). The equipment placement operator attempts to satisfy location preferences for

equipment items while also maintaining clearance requirements. The structural design operator seeks to ensure that the design of each component beam is structurally adequate and in conformance with AISC specification requirements. The construction critic operator attempts to expose any problem areas in the design by commenting on factors which will adversely affect construction cost.

The overall design is approached with the global objectives of minimizing total cost while providing a structurally sound and consistent solution. While these goals are fundamentally consistent with the local design goals, local "differences in opinion" on how to best achieve overall cost effectiveness require intervention and arbitration by the conflict resolver. Safety and structural adequacy follow automatically from the internal consistency maintenance features of FLEX.

### 5.2. Interdependences

The key to internal consistency maintenance lies in the knowledge which each component possesses about other members upon which it is dependent as well as other members which are dependent upon *it* Each beam, for example, "knows" who its parents are (the beams which support its ends). Each beam also knows which other beams and equipment loads consider it to be one of *their* parents. If a beam should be removed from the system, messaging between data objects permits its parents' child-beam lists to be updated automatically, thus maintaining consistency among spatial relationships. Similarly, if some attribute of a beam which affects its reactions should change (such as length, uniform load, child-beam reactions, applied equipment loads), the affected beam has the ability to recognize and act upon the need to recalculate its own end reactions, shear and bending moment and to submit to a `re-sign`. Beyond this self-adjustment, the beam can also notify its parents so that they can re-adjust accordingly.

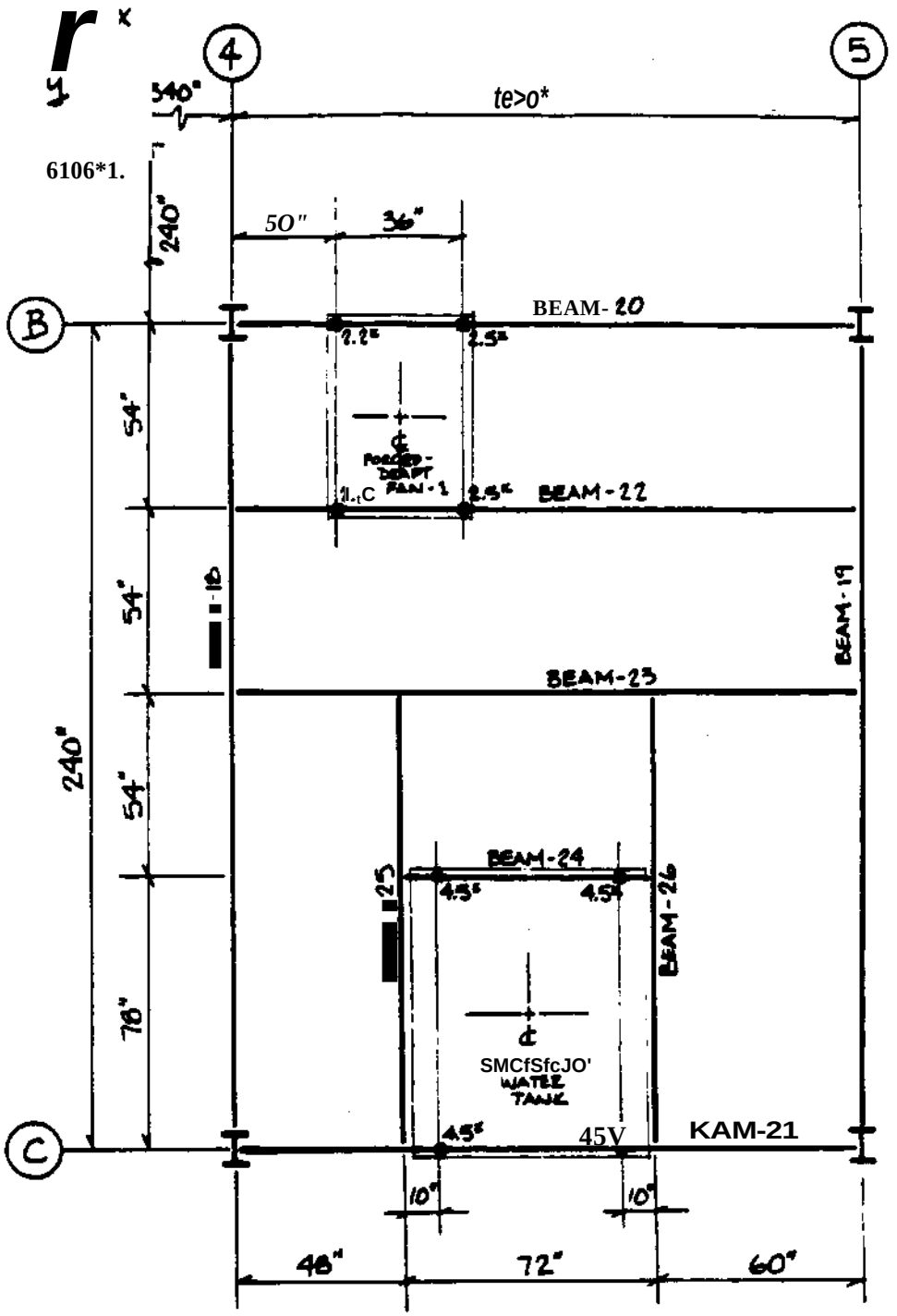
In similar fashion, each support load associated with a piece of equipment "knows" that it must be supported by a beam. Given a current load location, the equipment placement operator can search the beam list to determine whether a current load location is consistent with a current beam location. If so, the equipment load is associated with the beam's load-list (triggering the beam redesign responses described above). If no beam is found to be consistent with the current load location, the conflict resolver is invoked to investigate possible resolution options involving the adjustment of beam and equipment locations in various combinations, with total project cost minimization as the goal that will determine which resolution option prevails.

Figure 5-1 depicts a possible consistent design configuration for the FLEX prototype design example.

## 5.3. Implementation Environment

The implementation environment selected for the FLEX prototype is the KEE (Knowledge Engineering Environment) system developed by IntelliCorp. KEE is an appropriate choice for the illustration of several concepts which are central to the GUIDE problem-solving model:

- A frame-like data representation scheme is supported, enabling multiple data slots and facets to be associated with any data item. Procedural code and active values (demons) may also be attached, and inheritance is supported.
- The object-oriented paradigm provides direct messaging capability between data objects,




Rgur»5-1: Example of Consistent Floor Layout Solution

facilitating consistency maintenance strategies.

- The programming environment supports the creation of "multiple worlds" as an approach for the evaluation of hypothetical conflict-resolution alternatives.
- An integrated facility for production-system programming accommodates the development of the monitor module and design operators.

## 6. Typical System Interactions

This section presents several examples of system actions and responses in detail, in order to illustrate how the FLEX prototype implementation handles internally- and externally-enforceable maintenance situations. The descriptions of the various system actions are given in narrative form, but the terminology is consistent with the implementation environment; that is, "messaging- or "sending a message" refers to the object-oriented programming facility for communication between data objects, "activating" refers to the enablement of an active value, or demon, attached to a data object attribute, and a "method" refers to procedural code which is associated with a particular data object via direct attachment or inheritance. The process of "generating a load list" refers to a function whereby a beam can search its stored lists of child beams and applied equipment loads to dynamically generate a list of its currently-supported loads and their positions on the beam. The approach of creating this information procedurally whenever needed ensures that the resultant load list is always current.

### 6.1. Example: Adding an Equipment Load to an Existing Beam

The addition of a new equipment load to an existing beam represents a case in which the intelligent global solution database should be able to adjust automatically, since the consequences of such an action are predictable and unambiguous. Thus, this is an example of internally-enforceable consistency maintenance. A comparable sequence of actions would occur in the case where the value of an existing equipment load were altered. It is assumed in this example that the floor layout operator has already generated a configuration of support beams, perhaps after some previous interaction with the equipment placement expert. The current activity represents the addition by the equipment placement operator of a new equipment load at an existing beam location.

1. A description of the new equipment load is added as a new object in the database. This description includes a load identifier, location coordinates, load magnitude, units, and a parent-beam identifier. In addition, the new data object inherits active values and methods which trigger two activities immediately:
2. A consistency check is performed to ensure that the load location is coincident with the specified support beam. If it is not, then the parent-beam identifier is removed. If it is coincident, then a message is sent to the specified parent-beam to add this new load identifier to its stored list of supported equipment loads.
3. Upon modifying its list of supported equipment loads, the parent beam activates attached procedures to recalculate its end reactions and submit to redesign. When its end reactions change, these modifications in turn activate similar messages to *its* parent beams, and so on, until the change is propagated to the support points (columns).
4. For each beam whose loading is changed, a message is sent to the structural design

operator, requesting a redesign. Along with the request message, the structural design operator is given the information it needs to perform the redesign: the beam's maximum shear, maximum bending moment and maximum unbraced length. Each of these values is calculated by a method attached to the beam data object; to produce the information, the beam data object dynamically generates a new load list and invokes functions to calculate the design input values.

5. When the structural design operator has returned new beam designs for all affected beams, this information is stored by the individual beam data objects, and the global solution database has been restored to a consistent state.

## 6.2. Example: Locating an Equipment Load Where No Beam Exists

This example assumes that a floor beam layout has been established by the structural layout operator, and possibly some equipment locations have also already been established. Then, in establishing a position for a new piece of equipment, the equipment placement operator chooses a position such that an equipment support point is not consistent with any current beam location. The preferred resolution to such a situation depends upon the project priorities and may take any of several forms; there is no single, predefinable resolution strategy. Thus, the example is representative of a situation requiring externally-enforceable consistency maintenance, calling for the involvement of the conflict resolver.

1. As in the previous example, the information on the new equipment load is added as a new object in the database. The parent-beam slot is labeled *unknown*.
2. A consistency check, performed by the intelligent database, determines that there is no beam coincident with the load location in the current layout and flags the new equipment load location as *inconsistent*. This triggers a message to the monitor that the conflict resolver is required.
3. The conflict resolver's domain strategist recognizes the form of the conflict as a "no-beam" inconsistency and identifies three feasible conflict-resolution strategies:

    - Specify that the load location be modified to coincide with the nearest existing support beam location,
    - Specify that the nearest support beam location be altered to coincide with the equipment load in its current location, or
    - Add a new beam to the framing system to coincide with the equipment load in its current location.

    The conflict resolver can dictate the creation of alternate hypothetical worlds, each representing one of the solution alternatives. Each hypothetical world would be a permutation of the global solution database representing a consistent solution, complete with the global cost information which is routinely represented in the intelligent database. The conflict resolver's domain strategist would also have the capability to specify particular evaluation criteria for various conflict types occurring in the design domain. For example, it is a domain-specific characteristic of the floor layout problem that individual load locations are not independent: if you move one support leg of a tank, you must also move the other three. Therefore, in this type of a conflict, the evaluation procedure might logically consider two criteria:

- For a given conflict resolution alternative, how the choice impacts the global project cost, and
- For the alternative under consideration, how many *additional* inconsistencies are propagated in the solution.

The addition of the second evaluation criterion specifically for "no-beam" conflict resolution is an example of a domain- and situation-specific strategy to prevent solution "churning*.

4. The conflict-resolution alternative selected by the conflict resolver is imposed upon the global solution by messaging the appropriate design operator(s) to regenerate the affected portion of the solutions with the imposed values as new constraints.
5. Depending upon the conflict-resolution alternative chosen, further adjustments (such as beam redesigns) would take place within the intelligent database automatically, through its internal consistency maintenance system.

### 6.3. Example: Altering the Location of a Beam

The third example highlights a situation which is ostensibly a matter for internally-enforceable consistency maintenance but which also might have implications requiring external conflict resolution. In order to maintain spatial consistency of the floor framing system, it is necessary that connected beams *remain* connected when the floor framing plan is altered. In the intelligent global database of the FLEX prototype, the responsibility for maintaining parent-child consistency is delegated to each parent-beam as follows:

1. When a beam is moved (as determined by a change in its end coordinates), the beam notifies each child-beam framing into it that its location has changed.
2. Child-beams, in turn, invoke methods to adjust their end coordinates to maintain consistency with the new parent-beam coordinates.
3. The adjustment in length of child-beams triggers several consequences. Each modified child-beam recalculates its end reactions and design values and submits to a redesign, as in the first example. Further, the changes in end reactions impact the parent-beam, since these reactions represent loads on the parent-beam; therefore it adjusts its reactions and design values as well. In addition, the changes in beam end-coordinates and structural specifications trigger recalculation of the cost information on each affected individual beam as well as on the overall structural system.
4. If there are any equipment loads on the relocated parent-beam, the revised situation is examined to determine whether a consistent equipment support condition still exists. If not, the inconsistency is flagged, invoking an external conflict-resolution procedure as described in the second example.

## 7. Summary

The GUIDE model has been proposed as an approach for automating the cooperative multidisciplinary engineering design process in a framework that promotes solution consistency and value-based conflict resolution. A prototype system, FLEX, has been employed as an illustration of the GUIDE model in a

typical interactive design domain. It is the intent of the GUIDE model to draw upon previous development efforts in cooperative automated design, and to incorporate the value-based decision-making perspective exemplified by conventional engineering design practice. Two central aspects of the model address these objectives:

- **A flexible problem-solving framework** which preserves the modularity and autonomy of the individual bodies of expertise which contribute to the overall design effort while providing a choice of communication alternatives ranging from direct peer messaging to a formal control structure, and
- A high **degree of data** Intelligence, including intentional knowledge about the interrelationships and interdependencies among data items as well as value profile and design sensitivity knowledge which is essential to the rational decision process.

This synthesis of automated design concepts and conventional engineering management methodologies can elevate the usefulness of interactive computer-baseid design systems by combining "the best of both worlds":

- The high-speed computation and inferencing abilities of computer systems allow the exploration many more design alternatives than could be accommodated by manual approaches.
- The ability to evaluate alternatives with a view toward optimizing global project goals, and to resolve conflicts with the dual objectives of consistency *and design economy* increases the likelihood that system-generated designs will address the practical needs of the professional design community.