10

Artifact Configuration Across Different Design Levels

N. Murtagh
IMS Group, Information Systems Laboratory
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa 247, Japan
tel: +81-467-41-2846, fax: +81-467-41-2143
email: niall@ims.isl.melco.co.jp

Abstract

Design of artifacts goes through several different stages as the initial design description is broken down and refined in the process of searching for a solution. One of the important steps in the latter stages of design is configuration. This paper describes a flexible and easily maintainable system for configuration, using the layout design of an elevator shaft as a practical example. To effectively carry out the configuration task, it is necessary to simultaneously consider more than one design level, and to expand the model to include both numeric and symbolic processing. The system discussed here adapts previous constraint network ideas in order to automate the configuration reasoning. While the problem tackled here utilizes only one model of the design process, the possibility to apply the ideas to other models is pointed to.

Keywords

Intelligent CAD, configuration design, constraint-based reasoning, constraint networks

1 INTRODUCTION

Artifact design takes place at different levels as the design proceeds from initial description of requirements to ultimate realization. These levels of granularity usually correspond to different design stages, from conceptual or abstract design, through functional, physical part and sub-part levels, to parametric levels. Ideally each design level would be tackled separately and be completed before the next level is considered. However, in reality, design cannot be carried out in such a neat fashion. Inevitably, movement back and forth between design levels occurs, due to difficulties in fulfilling requirements, or to non-optimal decisions made at some earlier stage.

Hence, methodologies adapted specifically for one design level alone are not sufficient. It is necessary to consider two or sometimes more design levels simultaneously in carrying out a design, by using a mixture of strategies or by adapting one methodology to cope with more than one design level. Thus, design must involve not only reasoning in various *horizontal* planes,

T. Tomiyama et al. (eds.), *Knowledge Intensive CAD*

i.e., one conceptual level, but also in *vertical* planes, where movement between conceptual levels takes place.

The design of almost all types of artifact involves configuration, usually in the latter stages of basic and detail design. This research looks at a methodology for solving configuration problems utilizing a constraint-based reasoning methodology. Initially, it was intended to consider only one design level, focusing on numeric parametric design only. However, it became evident that consideration of parameters of other types was necessary in order to reach a solution. Hence, symbolic parameters were included in the configuration reasoning stage, requiring the adaptation of the basic methodology.

While the type of configuration problem tackled here utilizes one model of the design problem -- namely, that of parts described by parameters and variables -- the constraint-based reasoning ideas have the potential to be applied to other models, such as those focusing on functional design or even conceptual design. If some or all of the model can be considered as sets of entities and their relationships, the strategy discussed here has potential applicability.

This paper firstly refers briefly to the different design levels involved in artifact design, before outlining the type of configuration problem considered. It then discusses the use of constraint-based reasoning techniques and their application domain of elevator layout design. The advantages of using the techniques employed are explained together with difficulties encountered in obtaining a general methodology.

2 DESIGN LEVELS

The design of an artifact usually begins with a description of requirements from a customer or potential user of the artifact. This description will often be expressed in natural language or in some form not immediately amenable to technical specification. The initial step requires mapping the requirements to a technical specification. Once this is achieved, technical analyses can be used to break down the specification into various parts that can be handled either automatically by computer programs, or manually by human designers. The steps involved in moving through the design levels are summarized in Figure 1. Once design has advanced beyond the initial conceptual stages, much of the problem-solving involves the following:

- reasoning about entities and their relationships at a particular level or granularity,
- determining how to define those entities more precisely by breaking them down into constituent parts.

The stage or stages in design referred to as configuration deal with reasoning about how to arrange entities, and is the area this research considers.

3 CONFIGURATION DESIGN

Configuration is the task of organizing a set of known entities in a manner consistent both with required relationships among them and with externally imposed restrictions. Although the entity types are known there are generally various options for their realization. The problem can be considered as one of assigning values to variables while satisfying constraints on individual variables and between variables. Thus the configuration problem reduces to two principal steps:

1. The setting out of the problem description in terms of a declarative list of variables together with a set of constraints acting on those variables.
2. The selection of values for the variables consistent with the set of constraints.

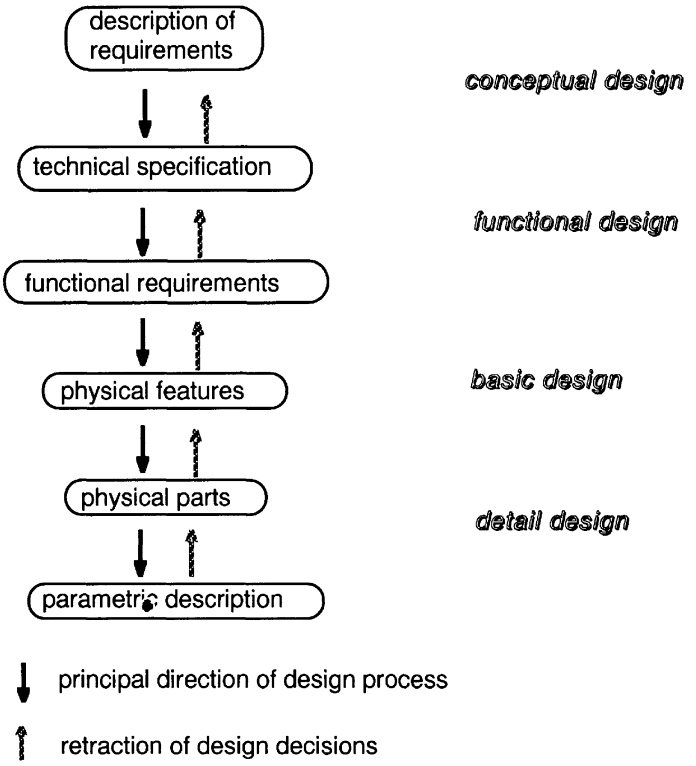


Figure 1 Principal Stages in the Design of Artifacts

Such configuration design is normally carried out by some generate-and-test method, involving iterative redesign either by human designers or by computer. However, conventional generate-and-test methods inevitably involve much redundant search of the set of possible solutions. Often, when the power of computers is available, simple generate-and-test or something close to it is used (Katsuyama *et al.*, 1991). This will clearly limit the type size of problems that can be considered -- e.g., NPT problems cannot be treated.

Here we apply techniques from constraint satisfaction research (Murtagh and Shimura, 1990). Constraint satisfaction provides a framework for efficiently solving search problems: it finds a consistent assignment for the parameters or variables, subject to a set of constraints on the allowed values. The techniques employed in constraint satisfaction can be used in the configuration design problem by considering as constraints both the required relations among the

parameters and variables, and the externally imposed restrictions on the possible values that may be assigned.

4 CONFIGURATION DESIGN OF ELEVATOR SHAFT

The design problem considered here is the selection and layout of mechanical and structural parts in an elevator, a form of configuration design. What makes this type of problem complex is the wide variety of part types and the variety of required relationships or constraints between the parts.

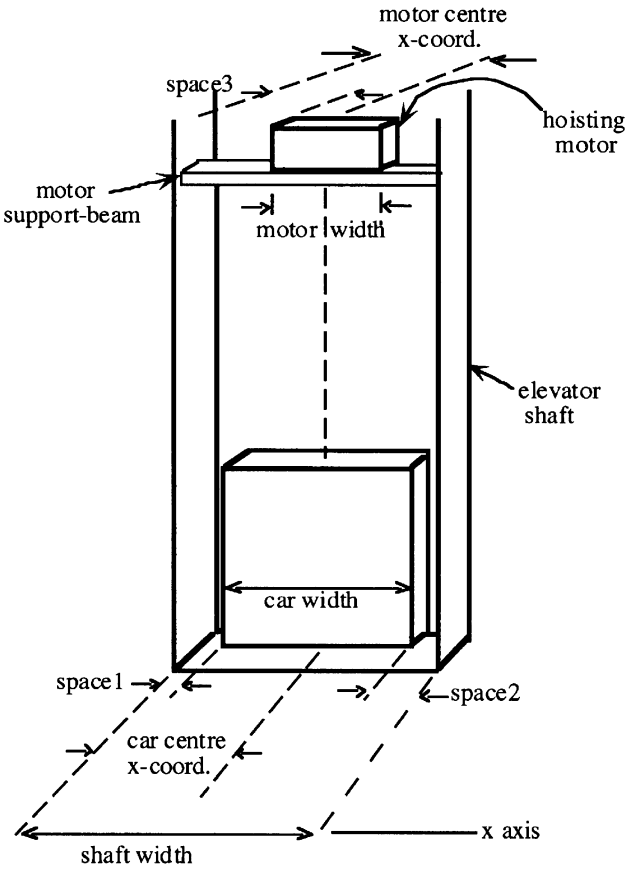


Figure 2 Application of Configuration Design: Layout of Parts in Elevator Shaft.

A simple illustrative portion of the configuration design of parts in the elevator shaft is shown in Figure 2. Generally, the elevator shaft size will be fixed before the selection of the car type and size, subject to spatial and other constraints. The car weight, in addition to other factors, is used to determine the hoisting motor type and size, and the motor must fit adequately in the machine room at the top of the shaft.

Conventional design methods have used generate-and-test strategies to configure the elevator shaft: the various parts were selected and then tests carried out to check for compatibility. Where a constraint violation or an inconsistency was found, the system would return to the selection step and generate a new configuration for testing. In order to avoid the redundancy and repetition involved in such a strategy, the system outlined here:

- selects and tests each parameter, before moving to the next one;
- when a test produces a negative result, uses intelligent back-propagation to steer the search for a solution.

Thus all the constraints acting on a parameter are considered when that parameter is processed, instead of taking groups of parameters (corresponding to a physical part, etc.) and assigning values to them before checking.

Using the present system to configure the elevator parts, a designer proceeds in the following way:

1. The problem is firstly described as a list of variables in a problem description file. The variables here cover all parameters in the artifact being designed. Domains of possible values are assigned to all variables.
2. A constraint solver is then run on the description, building a constraint network, before processing each variable. The variables are represented as nodes in the network with constraints or required relations linking them. Propagation of values then takes place: each node or variable is taken in turn and an appropriate value assigned to it, compatible with the constraints acting on the node. If no solution can be found with the current constraints and specified node domains, the designer is shown which node is not satisfied, in addition to a list of nodes which constrain the unsatisfied node. These are candidates for applying domain adjustment. Relaxing the domain constraints on one or more of these nodes will increase the chances of a solution being found. This process is repeated until a solution is found, or until no more adjustments are considered desirable.

5 CONFIGURATION ACROSS DIFFERENT DESIGN LEVELS

The design levels involved in finding a solution to the elevator shaft layout problem can be expressed in hierarchical fashion, as shown in Figure 3. Initially the automation of only numeric parametric design was considered, i.e., the lowest level in Figure 3. However, when certain numeric parameters could not be given values consistent with externally imposed constraints and those relating them to other parameters, it became evident that other design levels would have to be re-considered. For example, the dimensions of the hoisting motor are constrained by the dimensions of the car and shaft. When the motor dimensions are found to be inconsistent with required constraints the only way alternative dimensions can be found is by selecting another motor type. Such an alteration is not a numeric parameter change but rather a symbolic parameter one. Hence it requires at the very least, an expansion in the basic methodology used.

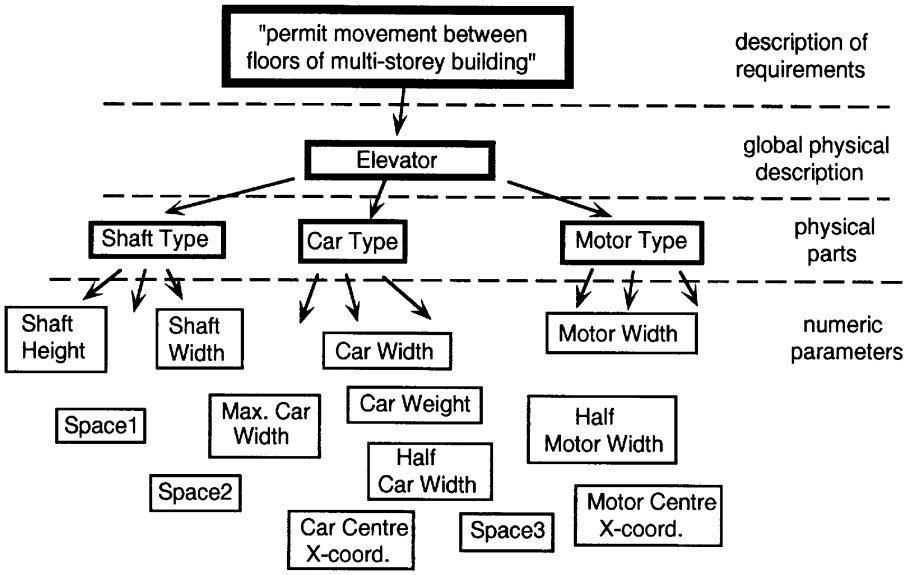
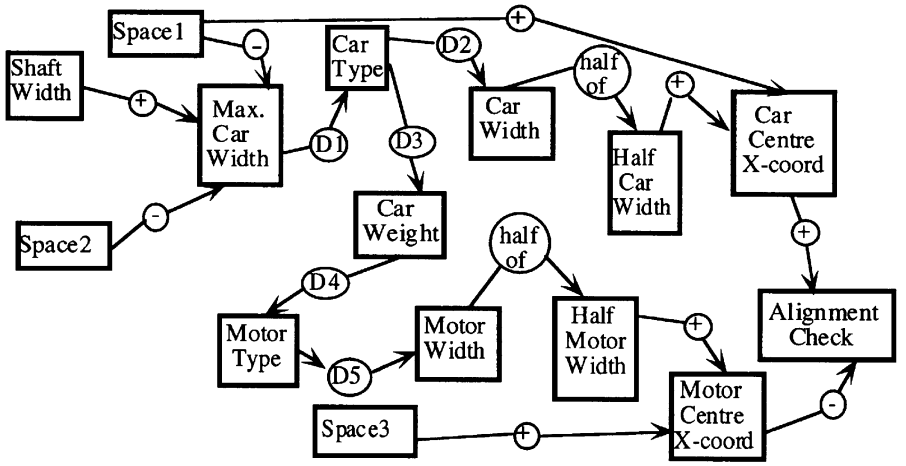


Figure 3 Breakdown of Design Artifact Description at Different Design Stages.



D1, D2, D3, D4, D5 refer to database retrieval:

Figure 4 Constraint Network Representation of Elevator Configuration Design.

The network used to model the selection and layout of parts is illustrated in Figure 4. Since it shows all dependencies between variables, it includes both numeric and symbolic valued nodes. Thus the "physical parts" level and the "numeric parameters" level (Figure 3) are combined in the net. As a result, the net-processing must be capable of integrating these two design levels.

The arithmetic constraints ("+","-", "half of") link numeric valued variables in a manner enabling forward and backward propagation to be efficiently carried out. Backward propagation requires heuristics to cater for network loops. (For details of the propagation mechanism see (Murtagh, 1991)). Non-arithmetic constraints link symbolic valued variables or combinations of numeric and symbolic variables, and require a different propagation mechanism. Examples of such constraints are those linking "Car Type" and "Motor Type" to other variables. These constraints are represented by D1, D2, D3 and D4 in Figure 4. In order to satisfy them, databases have to be accessed and appropriate values selected. Such database retrieval constraints are exemplified in rule format by the following:

if Car Weight = 200kg then Motor Type = type 1
if Motor Type = type 1 then Motor Width = 1.2 meters

Table 1 Database Retrieval Constraint Represented in Tabular Form

Node A	*Node B*	*Node C*
a1	b1	c1
a2	b2	c2
a3	b3	c3

These databases have been implemented in tabular form as in Table 1. Here, the 3-column database can represent constraints linking node A with nodes B and C (e.g., the relation between Car Type in Figure 4, and Car Weight and Car Width). Once the value of node A is known, the constraint linking it to node B will look up the appropriate database and find a value for B. A similar process is used for finding a value for C. This same database can also represent the constraints imposed by nodes A and B on node C, where the value of C depends on A and B. Thus a database-retrieval constraint must specify which column(s) is (are) the input value(s) and which column(s) is (are) the output value(s) for a particular node. Either exact or approximate matching can be carried out by the problem solver in selecting values from databases.

Design Description

The basic units for describing the design configuration problem are the variables or parameters which correspond to nodes in the constraint network. Each node is described by the following information:

Node name
Node type (soft or hard):
soft => value adjustable within allowed domain
hard => value not adjustable
Node domain type (continuous or discrete)
continuous domain => maximum, minimum and initial values must be given
discrete domain => name of data file containing domain values must be given

Other nodes constrained by this node
each forward-related node together with the constraint type is listed here;
-- any number of nodes directly constrained by a particular node may be listed.

The following shows a portion of the problem description, with 3 parameters or nodes, named *ShaftWidth, MaxCarWidth* and *CarType*. The first 2 have continuously valued domains with the maximum, minimum and initial values given, followed by the names of other nodes they constrain and the constraint type(s), e.g., node *ShaftWidth* constrains *MaxCarWidth* node with a + relation type. (+ implies that the value of the constrained node increases with the value of the constraining node.) The third node in the following list, *CarType* has a discretely valued domain which is contained in the file *datafileCarType*. This node constrains 2 other nodes in accordance with the relations specified by *D2* and *D3*. Further details of the implementation of the network are given in (Murtagh, 1991).

ShaftWidth hard continuous 5.0 1.0 3.0 MaxCarWidth + endOfNode
MaxCarWidth soft continuous 4.5 0.5 2.0 CarType D1 endOfNode
CarType soft discrete datafileCarType CarWidth D2 CarWeight D3 endOfNode

For nodes representing symbolic valued variables, the *Node domain type* is always *discrete*.

6 SYSTEM ARCHITECTURE

Figure 5 shows the main parts of the system architecture. The most important characteristic of the architecture is the separation of the problem description from the constraint definitions and from the configuration solver.

The 3 main parts of the system are as follows:

- The problem description is temporary, and only relevant for a particular problem. It can be edited at any time by the designer, e.g., if a solution is not found by the solver, the designer will normally adjust the description by relaxing some constraints, enlarging the allowable domain of some nodes/parameters, etc.
- The constraint definitions are part of the permanent knowledge base. Constraint-types specified in the problem description are restricted to those declared in this knowledge base. However, since the knowledge base is separated from the control mechanism, new constraint-types can be added as required, thus making the system applicable to a wide variety of problem types. Each constraint-type has an associated procedure which enables propagation of the constraint, i.e., a procedure which passes a node-value requirement between pairs of nodes linked by the constraint.
- The problem solver is permanent and may not be adjusted. Its function is to take the problem description, i.e., the list of parameters/variables and relations between them; then to form a constraint network; and finally to find a solution by attaching values to all nodes, consistent with the constraints.

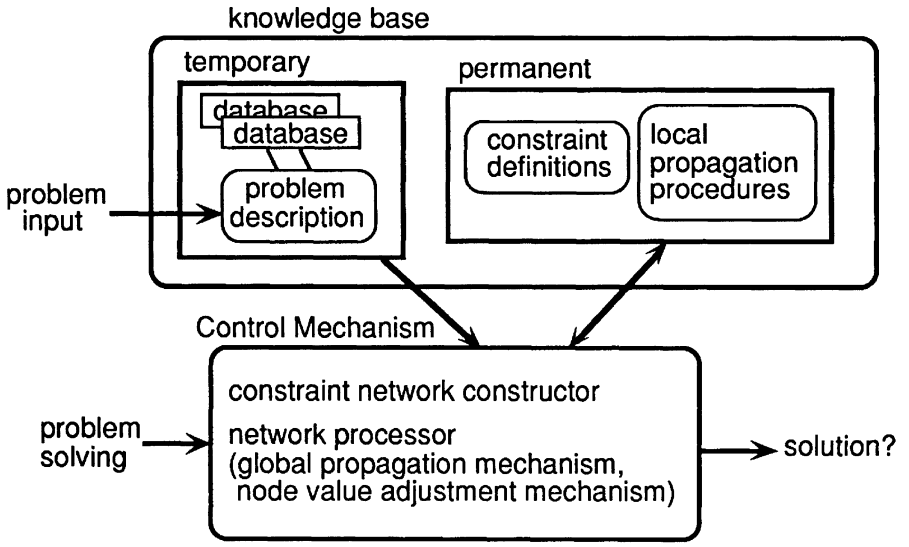


Figure 5 Architecture for Configuration Design Problem Solver.

7 IMPLEMENTATION

The system implementation gave consideration to the achievement of the following:

- Flexibility, in order to alter the design constraints, i.e., the ability to change both inter-node relations and required node domains.
- Maintainability, in order to add new parts or remove old ones, and thus the ability to develop the system incrementally.
- Portability, in order to use the system on various platforms and to permit automatic access to external databases.

For these reasons, C++ was selected as the implementation language, Since the problem-description is separate from the problem-solver, no re-compilation is necessary with new or adjusted problem descriptions. The only drawback is the necessity to define data types.

One of the main drawbacks with the system is the necessity to input all the node information and in the problem description file in accordance with the pre-defined set of possible constraint-types. This issue is currently being tackled by the provision of a graphical user interface. The interface is combined with a conventional CAD tool package, *I-DEAS (I-DEAS Master Series, 1995)*, enabling the user to combine the facilities of the graphic tool with the constraint solver. The input of part shapes and sizes is greatly assisted by the *I-DEAS* tool, and the constraints are input through a menu directly to the solver, which enables a wider variety of possible constraints to be considered.

8 RELATED RESEARCH

Various authors have discussed the ideas behind constraint networks for automated reasoning, (Mackworth 1977, Leler 1988) and applied to demonstration problem areas such as vision and combinatorial search (Montanari, 1974). The work outlined in (Murtagh and Shimura, 1990) illustrated the application of constraints to structural engineering design. The work presented in this paper extends these ideas to the problem of configuration design.

An earlier system for the layout design of an elevator system (Katsuyama *et al.*, 1991) separated the problem into a selection process of design parts, followed by the layout of these parts, thus necessitating a generate-and-test approach. The present system does not distinguish between selection constraints (mechanical, structural, etc.) and layout constraints (spatial, etc.), but considers all constraints in the same way. An earlier version of the current system for design configuration is described in (Murtagh *et al.*, 1992), focusing on the object-oriented programming aspects.

9 CONCLUSIONS

The aims of the present research were to produce a configuration design system that would permit the designer to alter the problem description and experiment with different part types in a design. A constraint-based approach was selected as being appropriate both to the type of configuration design considered and to requirements of the system -- flexibility and maintainability. The possibility of arranging the network as the user desires and of decomposing it into sub-problems provides considerable flexibility. The problem description can be altered easily: both constraints and domain values can be adjusted as required.

The advantages of using a network like this to represent variables or parameters and the relationships between them include the following:

- The piece-wise generate-and-test strategy implemented is more efficient than conventional global generation-and-testing since inconsistencies will be discovered at an earlier stage. In setting out the constraint representation, the user can arrange the more highly constrained nodes to be processed earlier, thus increasing the likelihood of inconsistencies being found early.
- The user can divide the problem up in any desired way by temporarily cutting arcs in the network, i.e., he can find a consistent assignment of values for a sub-set of nodes. He can then freeze these nodes so that they cannot be adjusted, and proceed to add new nodes to the problem. Thus the network approach facilitates problem decomposition.
- He can also choose to fix or relax constraints on any nodes, thus changing the direction of the search path.

In order to permit the type of reasoning necessary to enable the configuration to be carried out, not only numeric parametric design but also symbolic design had to be included. The combination of design levels was thus implemented. Various problems related to this mixing of both numeric and symbolic valued variables have been encountered, such as the implementation of an efficient backward reasoning mechanism to cater for redesign and adjustment of the current design description.

At the present stage of development, forward propagation involving both numeric and symbolic valued variables has been successfully tested, but the complexity of simultaneously realizing a generalized backward reasoning system suggests that the final version may partly separate the variable types. The issue reduces to a trade-off between the level of complexity that can be tackled and the robustness of the reasoning system.

Finally, the model utilized in this type of configuration design avails of the advantages of constraint reasoning. Other design models which also consider known or proposed entities and their relationships can make use of the strategies outlined. Work is also being carried out to determine which models are appropriate for these ideas.

10 REFERENCES

Dechter, R. and Pearl, J. Network Based Heuristics for Constraint-Satisfaction Problems. *Artificial Intelligence* **34**, 1987, 1-38.

I-DEAS Master Series 2.0, Student Guide, Structural Dynamics Research Corporation, 1994.

Katsuyama, T., Taki, H., Tsuji, H., Naito, A., Yoshida, M., Ohnishi, K. An Expert System for Elevator Design. *Expert Systems World Congress, Proceedings*, 1991, Pergamon Press, 38-45.

Leler, W. Constraint Programming Languages, their Specification and Generation. 1988, Addison-Wesley Publishing Company.

Mackworth, A. Consistency in Networks of Relations. *Artificial Intelligence* **8**, 1977, 99-118.

Montanari, U. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences* 7, 1974, 95-132.

Murtagh, N. Engineering Design Through Constraint-Based Reasoning, Ph.D. Dissertation, Tokyo Institute of Technology, 1991.

Murtagh, N., Shimura, M. A Parametric Engineering Design Using Constraint-Based Reasoning. *Proceedings of AAAI*, 1990.

Murtagh, N., Taki, H., Katsuyama, T. An Object-Oriented Constraint-Based Approach to Layout Design. *Japanese Society of Artificial Intelligence 6th National Conference*, 1992.

Taki, H., Katsuyama, T., Tsuji, H., Naito, A., Yoshida, M., Ohnishi, K. An Expert System for Elevator Design, *Operational Expert System Applications in the Far East,* 1991, Pergamon Press. 43-53.

BIOGRAPHY

Niall Murtagh carried out his graduate studies at University College Dublin and Tokyo Institute of Technology, where he completed his PhD in 1991. Since then he has been with Mitsubishi Electric, based in Kamakura, Japan. His research interests include the application of AI to engineering design utilizing automated and constraint-based reasoning. Recently he has been involved in the management of the Gnosis Project within the Intelligent Manufacturing Systems Program.