35

Information Support of FMS Configuration Design

Alexander V. Smirnov, Leonid B. Sheremetov, Pavel A. Turbin
Russian Academy of Sciences
St.-Petersburg Institute for Informatics and Automation
39, 14th Line, 199178, St.-Petersburg, Russia,
telephone: +7(812)218-80-71, fax: +7(812)218-06-85,
e-mail:smir@iias.spb.su

Abstract

This paper describes an approach to configuration design, based on an integration of dynamic constraint network and object-oriented paradigms for knowledge representation. The principles of the above concept are discussed. An implementation of the concept for information support of configuration design of FMS is considered on an example of environment for the Design of Structured Objects - DESO.

Keywords

Knowledge representation, dynamic constraint networks, object-oriented technique, configuration design, production systems.

1 INTRODUCTION

New market opportunities demand constant increase of product quality and decrease of its cost in rapidly changing environment. This trend has become more prevalent during 90's. As a consequence, the traditional production concept of stabilized systems is overshadowed by concerns with flexibility and competitiveness, realized in the FMS concept (Lepikson, 1995). In order to cope with these new paradigms, companies need to deeply transform their product development structure and this must be also accompanied by on-line transformations in the production system. The solution to the problem of rapid, flexible, and accurate manufacturing system design will come from knowledge-based Decision Support Systems, which enable

L. M. Camarinha-Matos et al. (eds.), *Balanced Automation Systems II*

fewer people to make rapid and better-quality decisions on manufacturing configurations from standard template solutions under constraints with reduced variance.

According to a general intelligent design theory introduced in (Ohsuga, 1989, 1991; Yoshikawa, 1987) configuring can be considered as the main content of structural analysis and synthesis of the production system. Typically, production systems have hierarchical structure which implies a hierarchy of constraints related to various levels and aspects of design (Freeman, 1990; Heinrich, 1991; Serrano, 1991; Smirnov, 1994; Tsang, 1991). Conventionally, constraint satisfaction is applied to problems with fixed set of constraints. However, the multi-aspect design of the production system put forward the problem of dynamic constraint satisfaction, that treats set of applied constraints in dependence with an aspect of design (Smirnov, 1994).

The goal of this paper is to introduce the concept of a knowledge based configuration design and to show how to implement the above paradigm using appropriate formalisms and means for intelligent data base processing.

2 PRINCIPLES OF CONFIGURATION DESIGN TECHNIQUE

2.1 Statement of a problem

Configuring is the construction of a complex system according to the requirements of a specification by selecting, parameterizing and positioning instances of suitable component types from a given catalogue (Heinrich, Jungst, 1991). We also include the creation of new component types using inheritance mechanisms into configuration procedure.

Let as consider the content of configuration design. We have the main goal and a number of requirements or specifications the target system being designed should meet. These requirements, described in terms of quantitative and qualitative variables, are mapped onto the attributes of target system entities depending upon an aspect of design. An aspect defines particular entity attributes to be taking into consideration according to the designer viewpoint on an object to design. Each entity is described by a set of attributes and functional relations, which connect attributes inside an entity and attributes of entities of different types. We distinguish between different types of functional relations representing rules and algebraic expressions. The design problem can be formulated in the following way: to compose the system from a number of template solutions to meet outside requirements. Hence, each entity, or designed object component, can be realized as an element from a set of template solutions.

A Dynamic Constraint Network (DCN) model seems to be the most suitable for this class of systems.

2.2 Axioms and heuristics of configuration design

In order to realize the above concept scheme in a computer environment, appropriate formalisation of the concept has to be introduced. A number of axioms and heuristics are at the background of a computer model. The following axioms are proposed:

- Each designed object consists of a finite set of components and connections between them.

- Connections between components are defined by a production process.
- These connections are defined by a finite set of attributes together with a set of rules of their consequence.
- Two components are connected with each other if the intersection of sets of their attributes is not empty.
- Each component has a finite set of template solutions.
- Each component (template solution) is defined by a finite set of attributes.
- Each component (template solution) attribute must belong to one of the following groups (aspects): technological, structural, operational, economical.
- Two solutions are consistent if corresponding components are connected and attributes of their connection are confirmed.

These axioms are complemented by the following heuristics:

- A set of attributes, defining a solution has a hierarchical structure.
- All template solutions are organized into taxonomies: class (component name) → type (concept of template solution implementation) → template solution.
- Each type defines a set of attributes of a template solution.
- Substitutable solutions are those of the same type with closest values of similar attributes.

2.3 A structured model of a designed object

The model of deductive reasoning of project solutions S, based on the above mentioned paradigm, can be formulated as follows:

$$S = \langle M, I \rangle, \qquad (1)$$

where I is a set of reasoning rules,
M is a structured model of a designed object, defined as follows:
$M = \langle R, Pr1, Pr2, Pr3 \rangle$,
where R is a set of external requirements,
Pr1 is a set of rules defining the types of template solutions for components,
Pr2 is a set of rules of template solutions consistency,
Pr3 is a set of rules for template solution preferences.

These rules define a set of external constraints (R), a set of constraints on entity attributes defined by external constraints (*Pr1*), constraints on interconnections between entities or components (*Pr2*), and constraints, defining ordering of template solutions (*Pr3*). External constraints are divided into integrated constraints, i.e. being applied to the components of an object to design after mapping according to *Pr1*, and local constraints, applied directly to an object attribute.

A structured model of a designed object M is represented as multi-layer dynamic constraint network (DCN) with hierarchical and associated relations, which can be defined in terms of multi-layer logic (Ohsuga and Yamauhci, 1985). A set of hierarchical relations includes 'part-of' and 'is-a' relations, according to axioms and heuristics, defined above. Associative relations represent functional relations, e.g. 'to be consistent' and 'to precede'.

An excerpt from constraint model of a problem domain is depicted in Figure 1, which shows connections between component types, and connections between attributes (a_i and b_j) for Machine Tool and Robot expressed by logic (b_1,a_2,b_3,a_3) and functional (a_3,b_5) constraints. This figure shows the way of mapping of relations between components, defined on the class level, into the level of attribute values consistency.

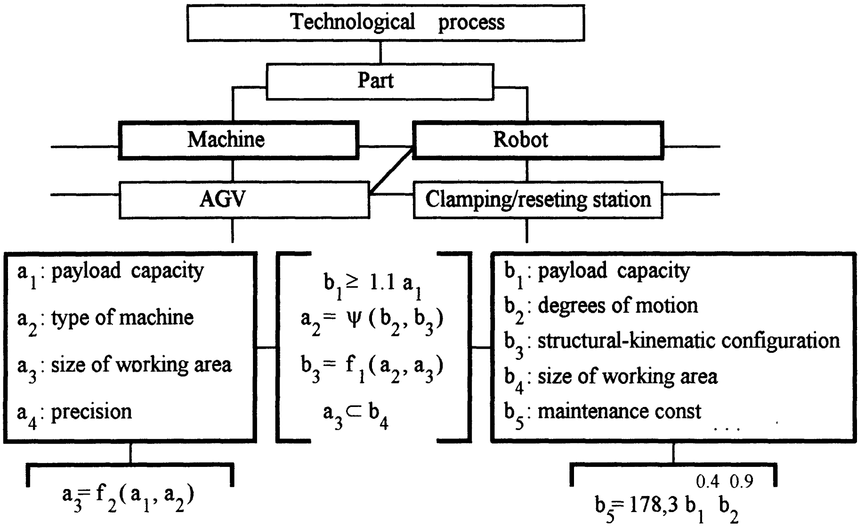


Figure 1 An excerpt from constraint model of FMS.

2.4 Deductive reasoning

General reasoning strategy is based on top-down methodology. The following sequence of constraints satisfaction is applied: project constraints → aspect constraints → structural constraints → qualitative constraints → quantitative constraints. Structural and qualitative constraints are derived according to deductive inference, quantitative - on the basis of numerical search. Constraint sequencing is based on literal ordering according to their place in the descriptive hierarchy and its importance for decision making. Each literal is associated with the depth of an object decomposition which is used to implement top-down strategy, using rules, defined in (Latombe, 1977).

2.5 Dynamic constraint network model

To deal with the concept scheme of configuration design process defined in terms of constraints, DCN model is applied.

A static constraint network (SCN) (V,*dom*,C) involves a set of variables $V = \{i, j, \ldots,\}$, each taking value in its respective domain, *dom*(i),*dom*(j),..., and a set of constraints C. Each constraint $c(i_1,\ldots,i_q)$ constraining the subset $(i_1,\ldots,i_q)$ of V, is a set of tuples, subset of the

Cartesian product $dom(i_1)\times\ldots\times dom(i_q)$, that specifies which values of the variables are compatible with each other.

A DCN $\mathcal{P}$ is a sequence of static CNs $P_{(0)},\ldots,_{(a)},P_{(a)},P_{(a+1)},\ldots$, each resulting from a change in the preceding one imposed by "the outside world". The task of finding a solution in a DCN is NP-complete, so a number of local consistency algorithms are proposed (Mackworth, 1977; Montanari, 1974). The most widely used are those achieving arc-consistency, which check the consistency of values for each pair of variables linked by a constraint.

Definition. A DCN is arc-consistent if for all i in V, $dom(i) \neq \varnothing$ and for all v_i in $dom(i)$, for all $c(i_1,\ldots,i_q)$ in C with i in $\{i_1,\ldots,i_q\}$, there exists $v_{i1}\ldots v_{iq}$ in $dom(i_1)\ldots dom(i_q)$ such that $(v_{il}\ldots v_i\ldots v_{iq})$ is a tuple in C.

Arc consistency is very simple to implement and it has good efficiency, described in (Montanari, 1974).

To apply this technique to the configuration design problem domain we consider entity attributes as variables and relations between them as constraints (Figure 1). In the case of multi-aspect configuration design the set of constraints, function domain and a set of attributes is a sequence of SCN each resulting from a change of an aspect of design.

3 KNOWLEDGE REPRESENTATION IN THE DESO SYSTEM

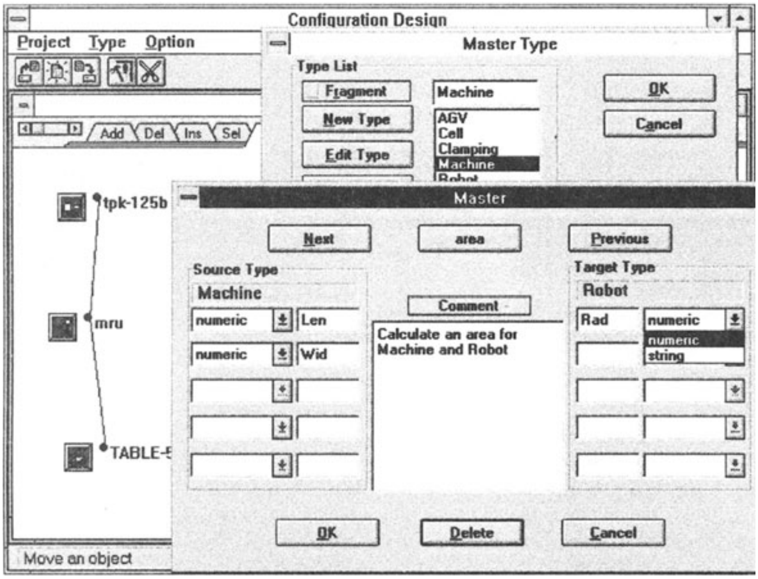
Let us consider the implementation of the concept, described above, for information support of a design process in the distributed environment for the Design of Structured Objects (DESO) to show how this paradigm is implemented to the FMS design problem domain.

The catalogue knowledge is the main knowledge base. It contains the knowledge about entity types (components) available for configurations. The ideal representation paradigm for the component types are classes of an object-oriented language, where the similarities between component types are used to construct a taxonomy with the catalogue components as leaves of the class tree..

The exception knowledge is attributed to the component types or classes it applies to, with different possible levels of expressiveness. Knowledge about simple incompatibility between components may be attached to a component (i) as a list of incompatible components checked against when a new component is selected or (ii) as predicate logic expressions about unacceptable partial configurations, treated as a method of checking procedure. It is possible to create an edge only if appropriate predicate expression checks that all the constraints are satisfied (Mackworth, 1977). The knowledge of this type defined in terms of integrated and local (atomic) constraints is stored in separate data bases.

In order to create supply a designer with the certain facilities of knowledge representation, master tools are implemented. A project expert tool is a compiler, which processes the rules defining integrated constraints. Each constraint can be associated with the precision value.

Figures 2 and 3 depict Master Function and Relation Definition tools. As shown in these figures the taxonomic structure of the component and resource type catalogues is exploited best with the familiar GUI tools allowing access to the knowledge interactively via class node icons. Functional constraints are organized as DLL library for functions, defined for object classes. Master Function mode is used to define a specific functional constraint for selected objects by associating object attributes with function arguments, as shown in Figure 2.



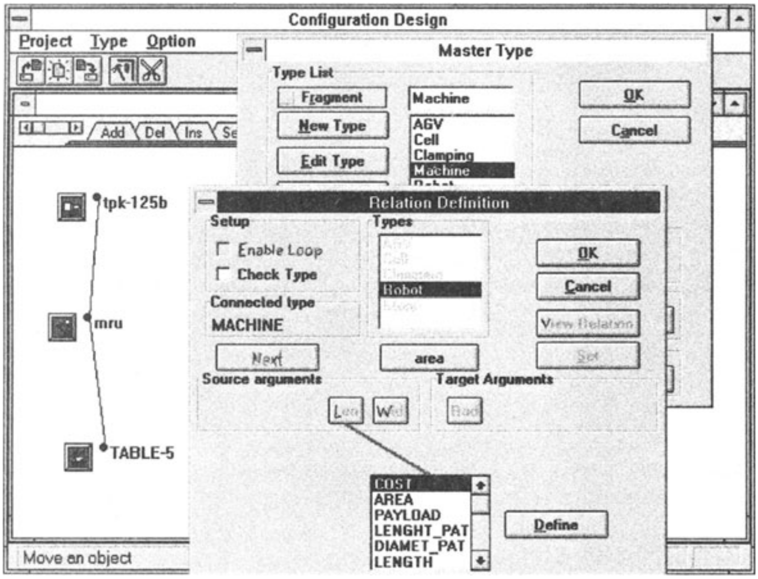
Type List: Machine
AGV
Cell
Clamping
Machine
Robot

Source Type: Machine
numeric Len
numeric Wid

Comment: Calculate an area for Machine and Robot

Target Type: Robot
Rad numeric
numeric
string

Figure 2 Master Function.



Type List: Machine
AGV
Cell
Clamping
Machine
Robot

Setup:
□ Enable Loop
□ Check Type

Types:
AGV
Cell
Clamping
Robot
Mere

Connected type: MACHINE

COST
AREA
PAYLOAD
LENGHT_PAT
DIAMET_PAT
LENGTH

Figure 3 Relation Definition Master.

4 CONCLUSION

We have considered an approach, which well do for configuration design of complex dynamic objects of different nature. The environment designed comprises means for group decision support for external requirements specification, project consistency control, and multi-aspect parallel configuration design of objects from predefined template components.

The universality of described knowledge representation scheme for all kinds of structured systems makes it feasible to provide powerful interactive tool for the knowledge base maintenance. The constraint-based approach has given us also an opportunity to realize "test-generate" programming methodology to prune problem search space that differs from "generate-test" approach, which is less efficient as compared with one we have applied.

Use of fragments takes advantage of distribution of design process among users, everyone working with a single fragment, which is then embedded in the object of the above level. In this case it is more natural to represent the knowledge of rational conceptual entities as a set of interacting autonomous agents processing local knowledge bases and to use the principles of agent-oriented programming systems as the basis for DESO development (Smirnov, Kulinitch et. al., 1995; Smirnov, Sheremetov et. al., 1995).

DESO environment has been developed under Windows 3.1 using Borland C++ 4.5 compiler, and IPX/SPX/NETBIOS protocol stack is used to support interaction between agents.

5 REFERENCES

Freeman, E.H. (1990) An Agent-Oriented Programming Architecture for Multi-Agent Constraint Satisfaction Problems, in *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, pp. 830 - 840.

Heinrich, M. and Jungst, E.W. (1991) A Resource-Based Paradigm for the Configuring of Technical Systems from Modular Components, in *Proceedings of 7th Conference on Artificial Intelligence Applications*. IEEE Computer Society Press, Los Alamitos, California, **1**, pp. 257 - 264.

Latombe, J.-C. (1977) AI in CAD System “Tropic”, in *CAD Systems* (ed. J.J.Allan), IFIP Working Conference in CAD Systems, Austin (Texas, USA).

Lepikson, H.A. (1995) Core Competence for Flexibility in Product Design and Manufacturing: One Approach for Long Term Competitiveness, in *Proceedings of the II International Conference on Concurrent Engineering: Research and Applications*. Concurrent Technology Corp., Washington, pp. 541 - 548.

Mackworth, A.K. (1977) Consistency in networks of relations. *Artificial Intelligence*, 8, pp. 99 - 118.

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

Ohsuga, S. (1989) Toward intelligent CAD system. *Computer Aided Design*, **21**, 5, pp. 315 - 337.

Ohsuga, S. (1991) Multiple Meta-Level Architecture and Man-Machine Interaction in a Knowledge-Based System, in *Industrial Applications of Artificial Intelligence*. Elsevier Science Publ. B.V., North-Holland, Amsterdam, pp. 67 - 75.

Ohsuga, S. and Yamauhci, H. (1985) Multi-layer logic - a predicate logic including data structure as knowledge representation language. *New Generation computing*, **3**, 4.

Serrano, D. (1991) Constraint-Based Concurrent Design. *Systems Automation: Research& Applications*, **1**, 3, pp. 217 - 230.

Smirnov, A.V. (1994) Conceptual Design for Manufacture in Concurrent Engineering, in *Concurrent Engineering: Research and Applications*. Conference Proceedings, Pittsburgh, Pennsylvania, pp. 461 - 466.

Smirnov, A.V., Kulinitch, A.S., Sheremetov, L.B., Romanov, G.V., and Turbin, P.A. (1995) DESO: A Constraint-Based Environment Prototype for Cooperative Design of FMS, in *Proceedings of the III IASTED International Conference*. Cancun, Mexico. IASTED/ACTA Press, Anaheim - Calgary - Zürich, pp. 384 - 387.

Smirnov, A.V., Sheremetov, L.B., Romanov, G.V., Turbin, P.A. (1995) Multi-Paradigm Approach to Cooperative Decision Making, in *Proceedings of the II International Conference on Concurrent Engineering: Research and Applications*. Concurrent Technology Corp., Washington, pp. 215 - 222.

Tsang, J.P. (1991) Constraint Propagation Issues in Automated Design, in *Expert Systems in Engineering: Principles and Applications* (eds. G.Gettlob, W.Nejdl), Springer-Verlag, Berlin, 462, pp. 135 - 151.

Yoshikawa, H. (1987) General Design Theory and Artificial Intelligence, in *Artificial Intelligence in Manufacturing*. Elsevier Science Publishers B.V., North-Holland, Amsterdam, pp. 35 - 61.

6 BIOGRAPHY

Alexander V. Smirnov is a head of computer aided integrated systems laboratory at St.-Petersburg Institute for Automatics and Information of Russian Academy of Sciences (SPIIRAS). His research interests include systems theory, artificial intelligence, concurrent engineering, design theory of manufacturing systems, and decision support systems. He received a Ph.D. degree from St.Petersburg State Electrotechnical University in 1984, and a Doc. Nat. Sci. from SPIIRAS in 1994.

Leonid B.Sheremetov is a senior researcher at SPIIRAS. His research interests include object-oriented programming, expert systems, and artificial intelligence, decision support systems. He received a Ph.D. from SPIIRAS in 1989.

Pavel A. Turbin is a researcher at SPIIRAS. His research interests include object-oriented programming, software engineering, information systems, and databases. He received a B.Sc. from St. Petersburg Institute for Precision Mechanics and Optics in 1995.