An extended version of this paper appeared in the ASME Journal of Mechanical Design 114(1): 187-195, March 1992

CONFIGURATION TREE SOLVER:

A TECHNOLOGY FOR AUTOMATED DESIGN AND CONFIGURATION

Alexander Kott, Gerald Agin
Carnegie Group, Inc.
Five PPG Place
Pittsburgh, PA 15222

David Fawcett
Electronics Division
Ford Motor Company
Dearborn, MI 48121

ABSTRACT

Configuration is a process of generating a definitive description of a product that satisfies a set of specified requirements and known constraints. Knowledge-based technology is an important factor in automation of configuration tasks found in mechanical design. In this paper, we describe a configuration technique that is well suited for configuring "decomposable" artifacts with reasonably well defined structure and constraints. This technique may be classified as a member of a general class of decompositional approaches to configuration. The domain knowledge is structured as a general model of the artifact, an and-or hierarchy of the artifact's elements, features, and characteristics. The model includes constraints and local specialists which are attached to the elements of the and-or-tree. Given the specific configuration requirements, the problem solving engine searches for a solution, a subtree, that satisfies the requirements and the applicable constraints. We describe an application of this approach that performs configuration and design of an automotive component.

1 INTRODUCTION

We describe a configuration technique that is well suited for configuring "decomposable" artifacts with reasonably well defined structure and constraints. This technique may be classified as a member of a general class of decompositional (abstract refinement (Mostow, 1984)) approaches to configuration. Other techniques within the same class of approaches have been utilized in (Brown and

Chandrasekaran, 1985), (Maher and Fenves, 1985), (Mittal and Araya, 1986), (Mitchell et al., 1985), (Preiss, 1980), (Steinberg, 1987).

This configuration methodology is intended for a weakly-connected, "nearly decomposable" (Simon, 1969) configuration artifact. Such an artifact can be subdivided into parts or characteristics with relatively weak interactions among them. Some of these parts or characteristics can, in turn, be decomposed into a number of sub-characteristics or parts, and so on.

Many mechanical products may be viewed as "nearly decomposable." This is particularly true for those stages or types of design where the designer has a clear picture of the product's composition, the design decisions that need to be made, the alternatives from which to choose, and the constraints that cause coupling between the decisions. These are the types of mechanical artifacts and the types of mechanical design processes for which this approach is applicable.

The decomposition model represents the configuration process as a sequence of steps, where each step starts with an incomplete configuration state and produces a configuration state of a greater completeness (in the sense that the new state contains more information about the configuration object) by adding to one of the components its more detailed description (either its specific "committed" implementation, or its decompositional description). The new configuration state's structure is the same as the initial state's structure at least at the level of abstraction found in the initial state (Kott and May, 1989).

The decomposition allows the configuration artifact to be viewed as a tree-like hierarchic structure. We call this structure the Configuration Knowledge Tree. The root of the tree is the final configuration artifact itself. The leaves are elementary objects, which are either predefined or are sufficiently simple to be configured by some predefined procedures. In general, each part of the tree may be configured in more than one way, and, correspondingly, may have more than one decomposition. Hence the generalized configuration artifact may be represented by an "and-or" tree (Figure 1).

An or-node often represents a design variable. The children of an or-node are the alternative values available for this variable. In the process of design or configuration, an or-node is given an assignment, i.e., one of the alternative values is given to this design variable. An and-nodes may represent an assembly of physical objects or an aggregation of design variables.

It is important to note that such a decomposition must be known a priori, before the configuration process begins.[Even though the decompositions may be generated dynamically in the design process (see sections 3.1 and 3.2), the knowledge for generating the decompositions must be available a priori.] Only in this case may the decomposition be useful for configuration purposes. In a decomposable configuration problem, all possible configurations of the configuration artifact are

implicitly known beforehand. However, the space of all possible configurations is usually very large and to find a configuration that satisfies a particular set of configuration requirements is a computationally explosive problem.

To guide the search through this large space of possible configurations we use two forms of domain-specific knowledge: Local Specialists and Constraints.

A Local Specialist is assigned to suggest (guess) the direction of the search when constraints are not able to provide the answer, or to guess which of the or-nodes should be assigned next when there are no fully constrained or-nodes.

A Constraint here is primarily a procedure that accepts values of one or more decision variables and returns the utility associated with this combination of the values. The primary role of the constraint is to evaluate how good or how bad is the given choice of the values for the decision variables.

These three main concepts, Configuration Knowledge Tree, Constraints, and Local Specialists, are discussed in further detail below.

2 CONFIGURATION KNOWLEDGE TREE

A strong hierarchic structure is inherent in many industrial products and systems. A typical piece of machinery can be decomposed into a number of major modules, such that connectivity between elements within each given module is much stronger than between the modules themselves. The major modules can be further decomposed into submodules, assemblies, subassemblies, parts, features, etc. Decision-making in designing or configuring such an object is also done hierarchically: first, some top level features are established, then a more detailed level of assembly drawings is developed, etc. Decisions made at a higher level of the hierarchy have major impact on the lower level decisions. The knowledge about such an object is well suited for representation by the hierarchic schema approach.

The hierarchic schema (frame) approach has been in favor with AI researchers since the 1960's ((Manheim, 1966), (Preiss, 1980)). It is a specialization of a fundamental problem reduction (decomposition) approach of AI. The problem reduction approach may be explained by observing that humans often solve problems by factoring them into smaller subproblems, then factoring these subproblems into even smaller elements, and so on until resulting subproblems are small enough to be solved by some simple means, such as using known solutions. This approach has been used successfully for a number of engineering design and configuration problems, where it is often referred to as a "refinement" method ((Stefik, 1981), (Maher and Fenves, 1985), (Mostow, 1984), (Mittal and Araya, 1986), (Steinberg, 1987)).

The task of configuring a piece of machinery is well suited for a problem reduction approach. A task of configuring a complete machine can be subdivided into

tasks of configuring its major modules (or major features); the task of configuring a major module can be decomposed into tasks of configuring its major assemblies; and so on until we reduce our problem down to a task of choosing between standard assemblies or parts.

Thus, to enable a computerized configurator to use the problem reduction approach, it must be given at least two categories of knowledge:

1. How to decompose a given feature or a module into smaller sub-features or sub-modules.

2. What alternative choices or implementation options are available for each feature or module.

This information can be conveniently stored in an "and-or" tree of hierarchic schemata. A schema is a description of a concept, or an object, that contains its attributes, associated procedures, and relations to other schemata. Each schema is a node in a network (in our case it looks more like a tree) of schemata. Relations between the schemata form the links in the tree. This tree of schemata holds the knowledge of the products offered by a particular manufacturer.

For our configuration problem we store in each schema either the parts and features that comprise the object (in this case we call that schema an "and-node"), or the alternative implementation options available for this object (in such a case we call that schema an "or-node"). For example, to describe an engine we form an or-node schema "engine" and include in it a number of alternatives - it can be a V6-ABC model, or V8-XYZ model, and so on.

On the other hand, a particular engine model may include a number of components or features - engine block, cylinder head, cam-shaft, starting system, etc. The schema that represents the particular engine model is an and-node. These sub-items in turn have various optional implementation, or sub-elements.

The combination of all these schemata forms an and-or tree (see Figure 1). Each time the configurator needs to configure a product, it can refer to this Configuration Knowledge Tree and find what alternative choices are available, and also how to subdivide the task of configuring the product into smaller tasks of configuring product's sub-items. Note that schemata can hold other information as well - prices, weights, engineering limitations, drawing numbers, applicable standards, manufacturing and scheduling information, pending price or engineering changes, and so on. In this way a Configuration Knowledge Tree can be closely integrated with the Sales Order System, Computer Aided Design, Group Technology and MRP modules of the Computer Integrated Enterprize.

This approach is predicated on model-based reasoning. In our opinion this approach fits closely the specific features of the configuration task that we discussed

above. The knowledge base in our approach has a high content of declarative knowledge, is highly structured, organized into a network of locally self-sufficient chunks, connected with explicitly defined relations. Such a knowledge base is advantageous when ease of maintenance is of significant importance. In addition, this organization of knowledge is natural for a hierarchically structured product; it has a promise of making the most from the existing systems and information bases. Furthermore, this organization of knowledge does not limit us to any single direction of decision-making: it can be either a bottom-up configuration process, or a top-down process, or a combination of both.

3 LOCAL SPECIALISTS

The role of the Local Specialists is to carry the domain specific heuristics that help to guide the search through the space of possible configurations (as represented in the Configuration Knowledge Tree). They are expected to make a guess about which assignment of the or-node should be tried first when constraints do not have enough information to provide the answer, or to guess which of the or-nodes should be assigned next when there are no fully constrained or-nodes.

In those design and configuration tasks that deal with hierarchically structured objects, there are significant advantages in organizing procedural knowledge as a hierarchy of local experts (Brown and Chandrasekaran, 1985). In our approach, Local Specialists are procedures or rules responsible for actually making the decisions about both the sequence of the configuration process and the selections among various alternative implementations/decompositions. Each of the local specialists is assigned to a single node (schema of an object) and contains only a limited amount of knowledge germane to that node. The specialists contain the bulk of the procedural knowledge available to the configurator. The maintenance of the knowledge is greatly simplified because

- the knowledge is organized into relatively independent "chunks" - specialists;

- each specialist deals with only one object and only at one level of hierarchy.

Because there are two kinds of nodes in the Configuration Knowledge Tree - and-nodes and or-nodes - there are also two different kinds of specialists - and-specialists and or-specialists.

The and-specialists are responsible for choosing an order in which parts of their respective and-nodes should be configured. Depending on the current state of configuration and on the constraints active for a given configuration session, this ordering can be different and may lead to a different outcome of the configuration process. In making its decisions, and-specialist refers to the previous decisions and to the constraints relevant to its node. All this information is entered into the rule or procedure of the specialist and then processed in order to obtain an ordering of the and-node parts.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.