From: AAAI Technical Report FS-96-03. Compilation copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Product Structure Based Reason Maintenance for Product Configuration†

Bei Yu
Department of Engineering, Glasgow Caledonian University
City Campus, Cowcaddens Road, Glasgow G4 0BA, UK
Email: B.Yu@gcal.ac.uk

Ken MacCallum
CAD Centre, University of Strathclyde
75 Montrose Street, Glasgow G1 1XJ, UK
Email: ken@cad.strath.ac.uk

April, 1996

Abstract

The aim of the process of product configuration in engineering design is to determine a set of elements and their relationships; i.e. to create an "arrangement" that satisfies design requirements and constraints. It becomes a difficult process when there are a large number of elements with various relationships and many design constraints. Current AI-based systems for product configuration have not been as successful as expected in their take-up by industry. This paper suggests that one reason for this is that the reasoning mechanisms in these systems are being developed without reference to product structures. Product structures on the other hand is a topic of increasing interest in both research in design and industrial practice in order to cope with a modern engineering design environment.

This paper presents a linkage between configuration process and product structures. The approach, i.e., a *Virtual Configuration Workbench*, is product structure based in contrast with existing approaches which are mainly rule driven or decision tree/network driven, integrated with an effective reasoning engine with a reason maintenance mechanism. A demonstration system called CONFIGMAN has been developed and illustrated briefly in the paper.

1 Introduction

In engineering design, any product or machine can be viewed as a technical system which consists of elements and their relationshipscite[Hubka 1980][Hubka 1988]. Configuration can thus be regarded as the process of creating an "arrangement" from a given set of elements by defining the relationships between selected elements that satisfies the requirements and constraints.

It is recognised that configuration is the part of design which is to do with *relations*. In configuration design, a set of elements and their relationships have been determined

[0]†This paper is submitted to AAAI 1996 Fall Symposium Series, November 9-11, 1996, MIT, Cambridge, Massachusetts, USA.

by the designer. Through this determination, the designer make decisions on selecting elements and defining relationships between selected elements.
However, configuration is not simply the design of a configuration; it is also the maintenance of configuration under change. The concept of configuration management pervades the whole life cycle of a product. Throughout the design and manufacture cycle, configuration changes may take place. After initial launch of a product, product versions and variants may be introduced. For some products, particularly large one-off capital goods, the product while in-service will undergo configuration changes. For many organisations, configuration management is a complex and time-consuming process.
Product structures which reflect relationships between elements can be complex. Even for routine design, it can be extremely difficult to configuration a new product structure rapidly and correctly. If the product needs innovation, configuration in addition becomes intermixed with other aspects of design.
In practice many products reuse past designs or elements. Since most products are changed depending either on their functionality or on particular requirements, such as the assembly requirements, the products are renewed incrementally rather than being changed totally to a new one. Reuse and adaption of previous products is very important in the design process.
Due to the complex product structure along with the complicated relationships between elements in a model of assembly, the configuration process faces several difficulties. Firstly, it is difficult to make decisions on selecting elements because of the unstructured or ill-structured information of the product domain. Secondly, it is difficult to remember all the relationships when the designer selects elements. When the designer makes decision on selecting elements from alternatives, he or she must consider those relationships which are related to the elements and design specifications. Thirdly, it is difficult to manage all the relationships and constraints during decision making. In other words, it is difficult to manage the consistency of configuration when the designer faces such design situations.
There is a trend towards exploring the research area of product structures because it has been recognised that the product structures have broad semantics during product development. Product structure is regarded as a way to present a product/product model[Janson 1995][Andreasen 1995], and appropriate product structures can offer a better understanding of product and design process. On the other hand, understanding of the product structures can help to build up a computer based design support model in a way that the product information can be used to create a design solution. Nonetheless, current research has shown that the configuration process in computer-based systems is being addressed without reference to product structures.

2 A Virtual Configuration Workbench

This paper presents a product structure based approach to product configuration, i.e., *Virtual Configuration Workbench.* Through this workbench, a designer can make decisions on selecting elements from the product knowledge base, based on the given requirements, and to put them together in a certain way as a configuration solution that meets the requirements and constraints. The configuration solution is represented by a set of product structures.
The central idea behind the approach is that product structures should be involved in the configuration process, and at the same time, an effective consistency maintenance

and a constraints handler assists designer to create a logical configuration model.
In the virtual configuration workbench environment, a designer can make decisions on selecting elements from the product structures based on the given requirements and constraints or can identify new elements if required, to produce a logical model for the physical configuration solution. This logical model is represented by a set of structures defined by a number of logical relationships between elements.
In this approach, the configuration process is regarded as decision making sequence towards the creation of a product structure. The configuration design process is illustrated in Figure 1. It starts from a product structure which presents the concepts of product parts breakdown in a logical way. This product structure implies somehow the relationships among product, module and elements families. The configuration process is carried out through a series of configuration decisions which consist of selecting elements, adding elements, deleting elements and configuring elements. Consequently, as a logical configuration model, the product structure is produced in which in the elements have been identified and the relationships between elements have been defined in a logical way. At the same time, a maintenance mechanism is applied during decision making to maintain the consistency among selected elements and decisions for ensuring the correctness of the configuration solution represented by a logical structure.

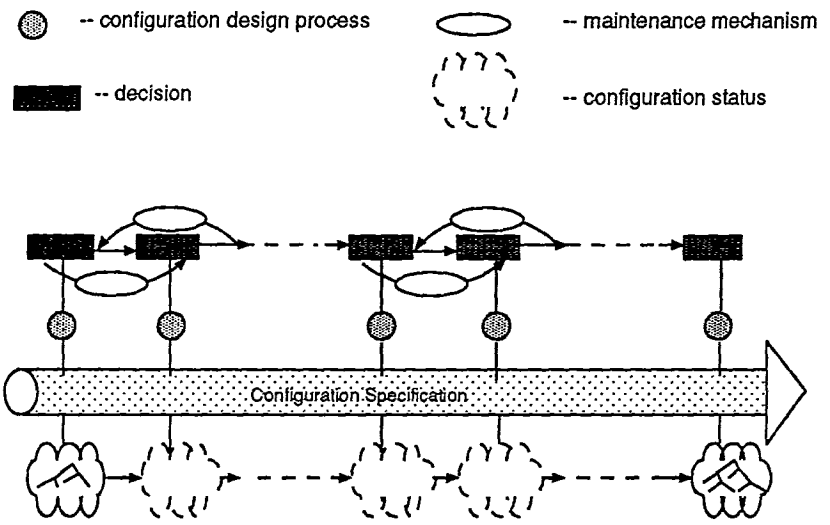


Figure 1: The Configuration Process in the Virtual Configuration Workbench

This configuration process is a non-monotonic process. During the configuration process, decisions could be changed for some reasons, and these changes are maintained by a consistency maintenance mechanism.

3 Developing Product Structures

Two interrelated product structures, that is, *Product Breakdown Structure (PBS)* and *Product Family Classification Trees (PFCTs)*, have been developed for the configuration workbench. The framework of the product structure formalism is shown in Figure 2. The PBS represents a conceptual product model in a hierarchy of a parts breakdown. Links in the structure are viewed as "a part of". Each node in this structure stands for a module, a subsystem or an element. All the attributes, features and properties of selected elements are recorded in the elements list. Each node in the PBS belongs

to its own family classification tree also called Product Family Classification Tree, in which a group of elements have highly commonality of functions, characteristics or manufacturing process. Nodes in a PFCT have relevant parts breakdown at different levels of detail.

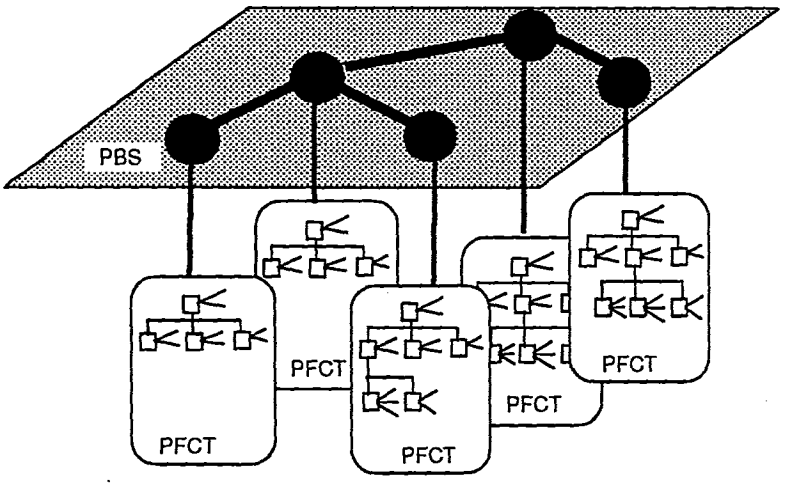


Figure 2: The Framework of the Product Structures

A product range can be classified as a set of PFCT, i.e., PFCTs, each of which is the tree structure that presents a class of product and its modules from an abstract level to product instances. Each node in the tree represents a product or a module class with its parts breakdown. The ancestor-descendant relationship between two classes is presented as "a kind of", i.e., a class of product is a kind of the superclass of product. There will be several product family classification trees which are related to one another. In other words, all existing modules, parts and elements that might be configured can be found in the given knowledge sources in terms of their own Product Family Classification Trees. In addition, constraints about elements exist among those Product Family Classification trees.
The criteria for classifying PFCT is the key to develop such a product structure. There are number of possible criteria for developing PFCTs, such as function, user requirements, product performance, application aspect, customer group, manufacture location, etc. For a product range therefore, there can be multiple PFCTs corresponding to different classification criteria[Yu 1995a].
Links in the structure are viewed as "a part of". Each node in this structure stands for a module, a subsystem or an element. All the attributes, features and properties of selected elements are recorded in the elements list.

4 A Reason Maintenance Mechanism

The problem solving method in this approach is shown in Figure 3. During configuration design, each decision can be shown as an individual consistent statement that relates to the previous decision. Whenever a decision is made, the previous dynamic constraints will be tested to check either if they need to be relaxed or if new constraints are propagated.
In addition, fixed, given constraints might be added depending on inference from previous decisions. On the other hand, the consistency among the decisions, and among the selected elements is maintained to ensure the validity of the decision sequence based

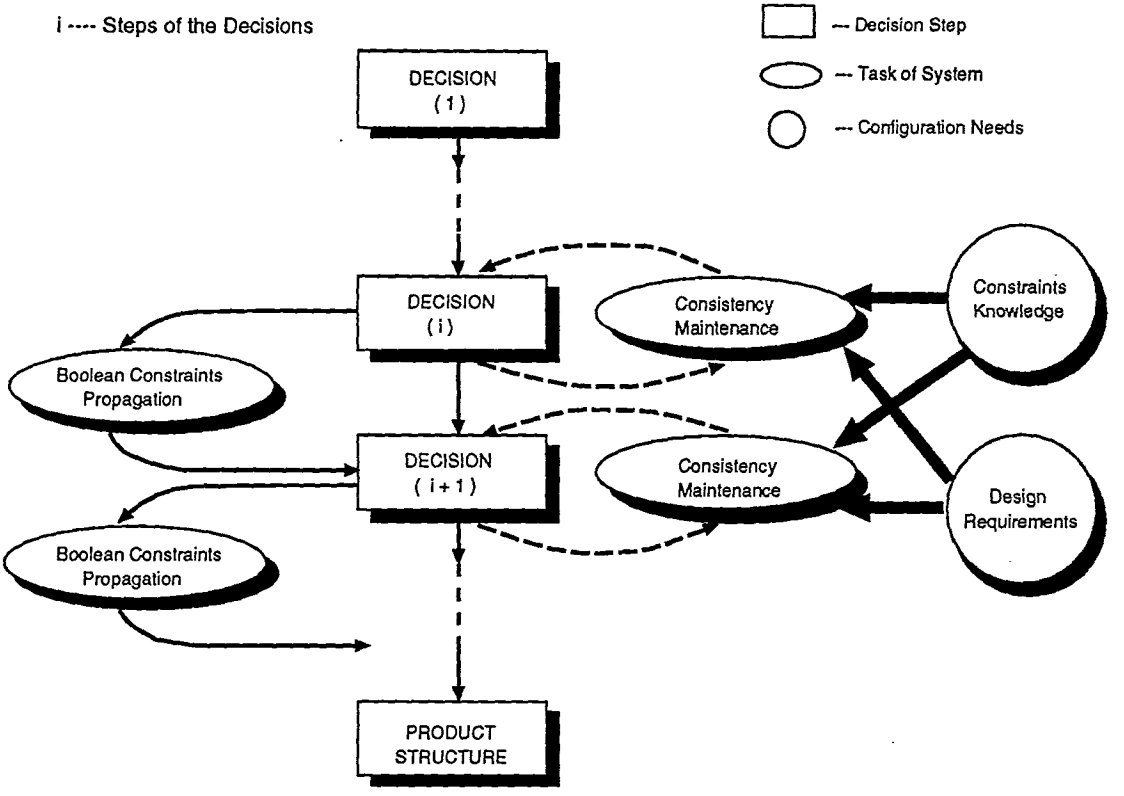


Figure 3: Consistency Problem Solving Method

on the constraints knowledge and design requirements. The process continues until the decision sequence satisfies all the requirements and constraints given by the designer. This problem solving mechanism is to assist the designer to create a product structure as a configuration solution during configuration design. The searching and mapping mechanism is applied under the PFCTs and the PBS product structures to generate a product structure in terms of configuration solutions. An instance of the PBS is created by fitting the selected elements into the PBS. The relationships between the selected elements are then defined through mapping on the PBS which is used as a concept tree in the knowledge base.

A reasoning engine has been designed to drive the overall performance of the configuration workbench[Yu 1994]. The inference engine is composed of three units: *a Decision Processor, a Logic-based Truth Maintenance System* and *a Forward Checking algorithm.* In this structure, the decision processor can be regarded as a problem solver which manipulates the configuration knowledge sources, guides the search, and creates the configuration solutions based on a series of decisions. The Logic-based Truth Maintenance system (LTMS) is used as the decision processor subsystem, to assist the decision processor to maintain consistency and dependency with change, as well as to handle logical constraints. The Forward Checking (FC) algorithm is applied for constraints pre-processing in order to increase the power of the inference engine in constraint management.

4.1 Decision Processor

As a problem solver, the task of the decision processor is concerned with the manipulation of the configuration knowledge sources and the maintenance of consistency and

dependency, either when a decision is made or when a decision is changed. In addition, it supports the configuration task to reach a solution in terms of a PBS instance by checking if selected components meet the requirements and constraints.
The decision processor can call the LTMS to maintain consistency among the decisions if necessary. In order to cope with the nonmonotonic configuration process, a nonmonotonic reasoning mechanism is used for maintaining consistency effectively. Specifically this mechanism is used either when conflicts appear after a decision is made, or when there is a contradiction in the updated situation in which case a decision needs to be changed. It maintains consistency dynamically i.e. it maintains the updated consistent situation for each step of the configuration process.

4.2 LTMS Unit

The LTMS[McAllester 1980][McAllester 1990] performs propositional deductions from a set of propositional premises which take the form of assignments of truth values to assertions. The constraints in the LTMS are derived from basic axioms of propositional logic. In other words, clausal constraints can be generated by each of the associated axioms of logical symbols *and, or, not* and *implies.*
The LTMS maintains a table of decisions and justifications for the decisions. It deduces the new justification for a made decision in order to constrain the further unmade decision. Meanwhile, it checks for consistency of a decision, that is, revises the beliefs that cause contradiction in the made decisions. The Dependency Directed Backtracking algorithm is used to identify and retract the decisions that underlie contradiction.
In addition, the decision processor with the LTMS unit has an ability to record decision sequences. It stores all the events of the configuration process and keeps the history of the cases of configuration design. Furthermore, it enables the designer to find the reasons for decisions i.e. the explanation facility in the inference engine.

4.3 FC Algorithm

As a constraint-based reasoning technique, the Forward Checking algorithm is used to increase the power of the inference engine by constraint management. The algorithm eliminates incompatible branches early on in the search so that the efficiency is improved compared with chronological backtracking. It performs a search process in such a way that at any given node in the search tree, its variable is bound to the next untried available value from the domain to check for compatibility. This value is checked for compatibility with each available value for each member of the future set, and any value which is incompatible becomes eliminated. Therefore each selected value is always compatible with the current binding of every member of the past set[Shanahan 1989].
In the configuration problem, the decisions can be shown as variables and each choice of decision can be viewed as a value in the option domain. The relationship between decisions can be presented as a set of constraint pairs. The task of the Forward Checking algorithm in configuration is to eliminate those options which are incompatible with the given constraints, so that the boundary of options for the solution has been reduced.

5 A Case Study

A demonstration system of the virtual configuration workbench called CONFIGMAN has been developed under the POPLOG AI programming environment[ISL 1985], and

can be found in[Yu 1995b][Yu 1995c]. As a case study, a worked example, i.e., configuration design of refrigeration plant controller from the ELM company based in Glasgow, has illustrated the feasibility of the overall approach.
The PBS and PFCT of the plant controller are shown in Figures 4 and 5. During the configuration process, a designer starts from the PBS of the plant controller as a conceptual tree, to select/identify elements from the relevant PFCTs of controller and its parts families based on the requirements, and to create a logical configuration model as a configuration solution. In Figure 6 for example, a partial configuration solution is shown. Notice that this may not be a final configuration solution; the system allows the designer to change decisions which have been made. If this happens, the reason maintenance mechanism is able to maintain the consistency among selected/identified elements and decisions corresponding to the changes.

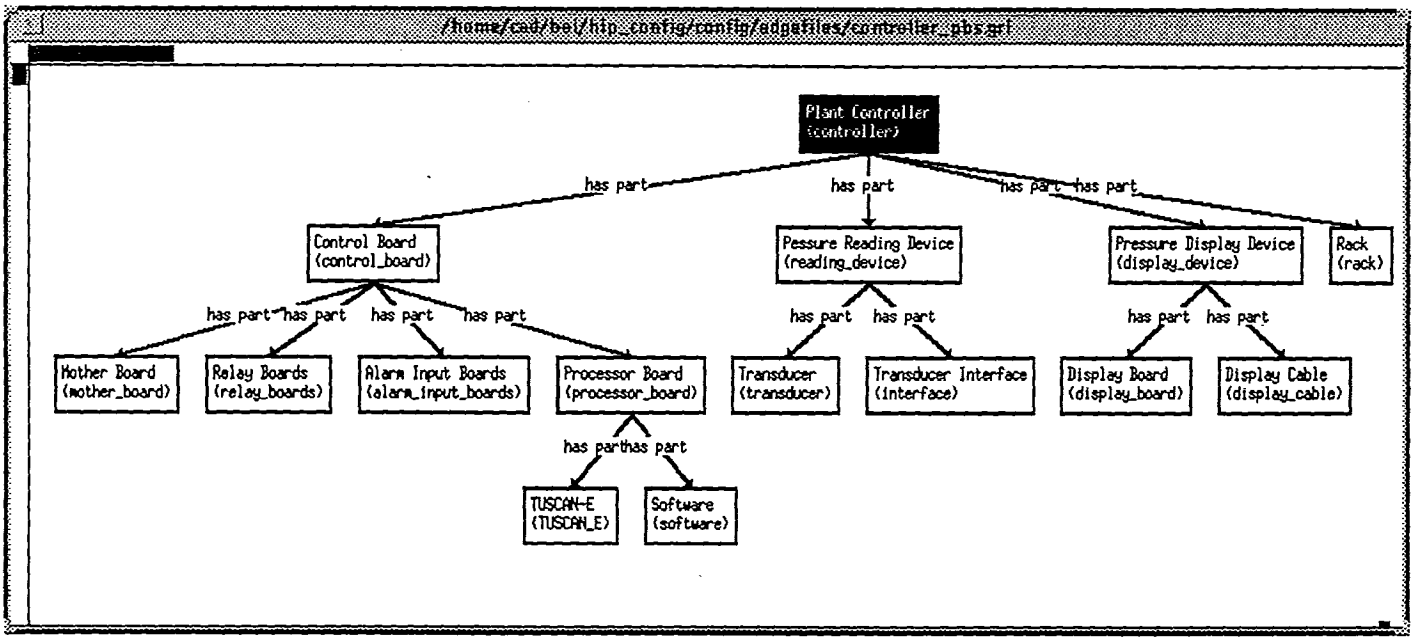


Figure 4: Plant Controller PBS in Display

6 Discussion and Conclusion

The approach presented in this paper is a product structure based configuration support approach. It is different from others in which the configuration process is either a rules driven, a constraints driven, or a decision tree/network driven process. It has been realised from this distinguishable approach that product structuring is closely related to the configuration process. The significance of this approach is to build up the relationship between the product structures and the configuration process, so that the hypothesis that the product structures can influence the configuration process has been verified. Meanwhile, this work paves the way for a better understanding of the configuration design process and product structuring, as well as the relationship between them.
For supporting configuration design, this approach offers a very flexible process procedure so that the designer could have more freedom during the configuration process. In other word, the designer can choose the way of configuring rather than following the

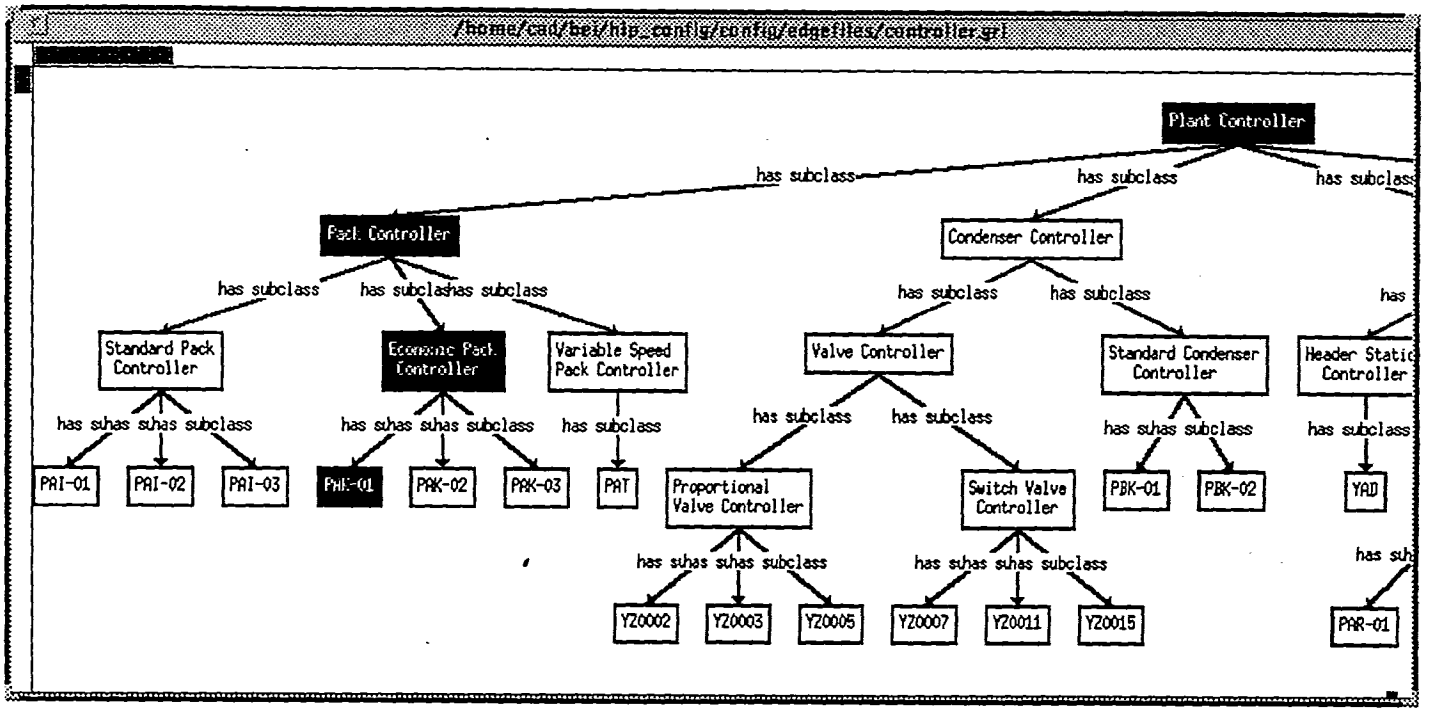
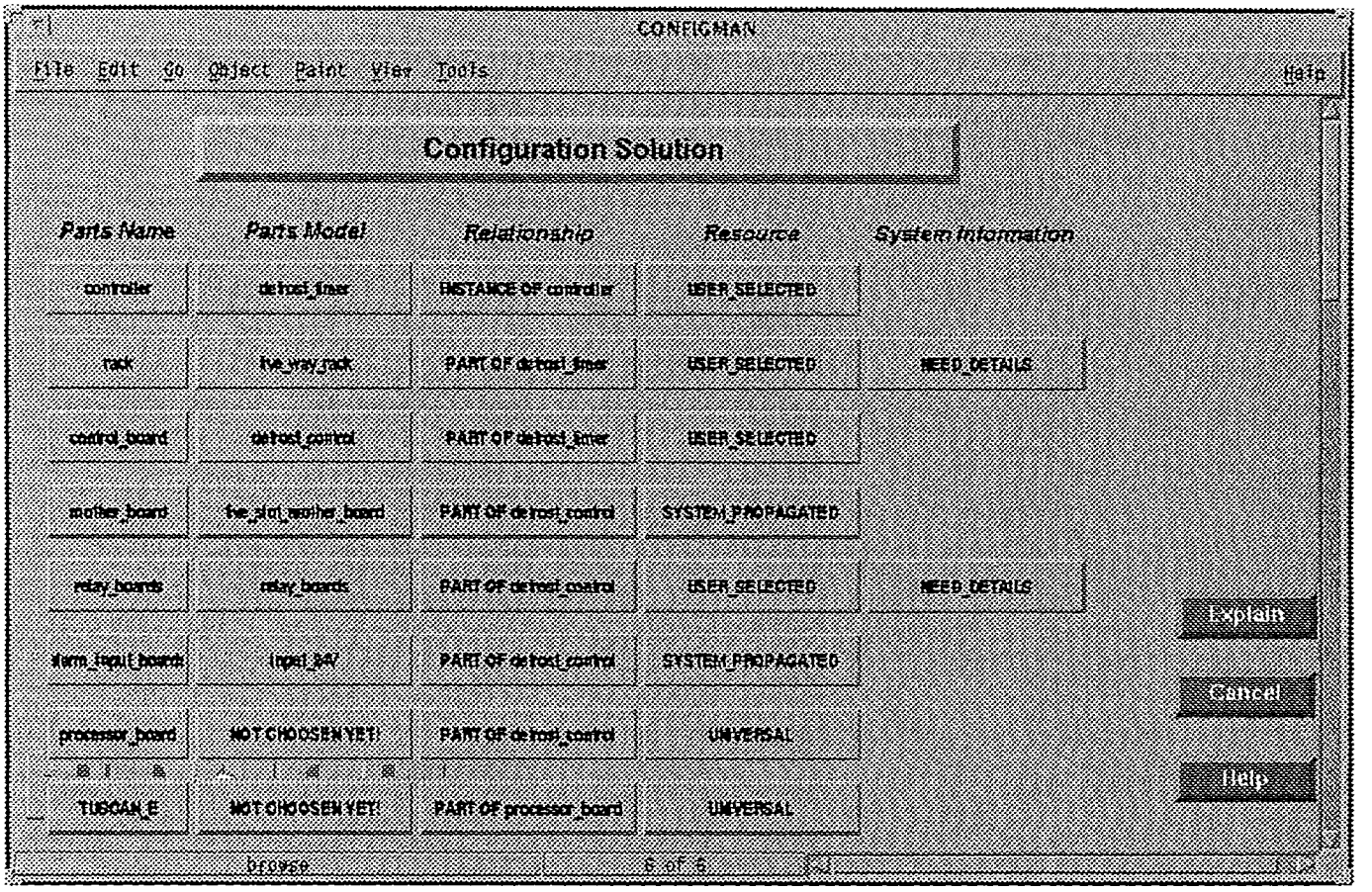


Figure 5: Part of the Controller PFCT

routine which is given by the system. Compared to other approaches which restrict the configuration decisions into a set of rules or decision tree/network, this approach does not prevent the designer to choose a strategysuch as either top-down, bottom-up or both. In other words, the configuration decisions are not limited a predefined decision order such as a decision tree or rules.

In order to support configuration maintenance effectively, a RMS problem solving approach has been developed. The need for using this approach comes from the highly interactive nature of the configuration support system. In the composition and qualification abstraction levels, at any point in time if the system does not have enough information to decide which element to pick, the relevant specifications which are formed as a set of propositions (propositional constraints) in the LTMS are required from the designer. The designer can choose any of these specifications in any order. The system will restrict the allowable answers for other specifications based on the evaluated specifications, i.e., LTMS runs the propositional constraints propagation. This is a specific feature of LTMS which other TMSs do not have. LTMS is able to express a local constraint as a boolean constraint, as well as to increase computational speed reasonably since any boolean constraint can be expressed in a LTMS and the constraint propagation is fast[McAllester 1990].

The constraints based reasoning technique enables the decision processor to be able to handle different logical constraints. The Forward Checking algorithm is able to reduce the boundary of decision selection to decrease the complexity of consistency maintenance and avoid unnecessary mistakes. In addition, the decision processor handles the local constraints propagation for constraining the next decision. The task of the Forward Checking algorithm is concerned with handling constraints before any decision is made. The benefit from this mechanism is to decrease the complexity of reasoning systems by problem decomposition, so that the constraint management ability in the inference engine is increased.



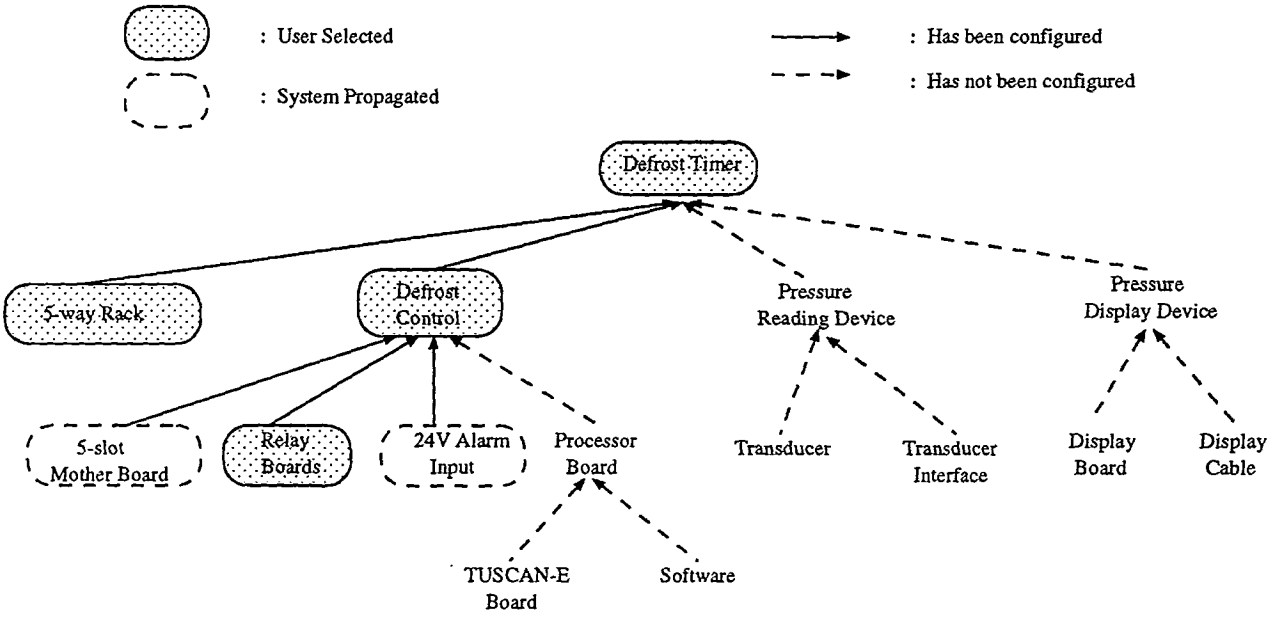


Figure 6: PBS of the Defrost Timer (Partial Configuration Solution)

In short, the paper presented a configuration support approach which is based on product structures. The conclusion derived from this work is that in a computational design support environment, product configuration process with reasoning system cannot be generic and self-governing; it must be linked to product structures. With appropriate product structures and an effective reasoning mechanism, products can be configured rapidly and correctly.

References

[Andreasen 1995] Mogens Myrup Andreasen, Alex Duffy, Niels Henrik Mortensen, **Relation Types in Machine Systems.** *Proceedings of the WDK Workshop on Product Structuring,* pp83-98, Delft, The Netherlands, June 22-23, 1995.

[Barrett 1985] R.Barrett, A. Ramsay and A. Sloman, **POP-11: A Practical Language for Artificial Intelligence.** Ellis Horwood Limited, ISBN 0-85312-940-1, 1985.

[Hubka 1980] Vladimir Hubka, **Principles of Engineering Design.** Butterworth&Co. (Publisher) Ltd., ISBN 3-85693-018-3, 1980.

[Hubka 1988] Vladimir Hubka and W. Ernst Eder, **Theory of Technical Systems.** Springer-Verlag Publisher, ISBN 3-540-17451-6, 1988.

[ISL 1985] Integral Solutions Ltd., **Poplog User Guide - POPLOG Version 15.0.** 1995.

[Janson 1995] Leif Janson, **Business Oriented Product Structures.** *Proceedings of the WDK Workshop on Product Structuring,* pp9-26, Delft, The Netherlands, June 22-23, 1995.

[McAllester 1980] D.A. McAllester, **An Outlook on Truth Maintenance.** *AI Memo 551, MIT AI Laboratory,* 1980.

[McAllester 1990] D.A. McAllester, **Truth Maintenance.** *AAAI-90 Proceedings,* July 29 - August 3, pp1109-1116, 1990.

[Shanahan 1989] M. Shanahan, R. Southwick, **Search, Inference and Dependencies in Artificial Intelligence.** Ellis Horwood Limited, ISBN 0-7458-0488-8, 1989.

[Yu 1994] Bei Yu and Ken J. MacCallum, **Reason Maintenance for a Configuration Management System,** *Expert System'94 Conference Proceedings: Research and Development in Expert Systems XI,* SGES Publications, pp173-185, ISBN 1-899621-01-6, 12-14th December, 1994.

[Yu 1995a] Bei Yu and Ken MacCallum, **A Product Structure Methodology to Support Configuraiton Design.** *Proceedings of the WDK Workshop on Product Structuring,* pp83-98, Delft, The Netherlands, June 22-23, 1995.

[Yu 1995b] Bei Yu and Ken J. MacCallum, **Decision Support for Configuration Management in Product Development,** *The 3rd International Conference on Computer Integrated manufacturing (ICCIM'95),* Singapore, World Scientific, pp278-286, ISBN 981-02-2973-9, 11-14th July, 1995.

[Yu 1995c] Bei Yu and Ken MacCallum, **Modelling of Product Configuration Design and Management by Using Product Structure Knowledge.** *Proceedings of the IFIPWG5.2 Workshop Series on "Knowledge Intensive CAD"*, Helsinki, Finland, September 26-29, 1995.