Paper presented at

THIRD INTERNATIONAL CONFERENCE

ilce'95

Concurrent Engineering & Technical Information Processing

January 30 - February 3, 1995

Espace Saint-Jacques
Paris

A New Approach to Product Configuration

Hans Jørgen Skovgaard
Research Manager
Beologic A/S
Hørkær 12 A, DK-2730 Herlev, Denmark
Tel. +45 4453 8888
Internet sko@beologic.dk

Abstract.

This paper describes a new method for product configuration based upon constraint techniques. The method is implemented in the Beologic Product Configurator **salesPLUS**. It has very strong features for the end user; free order of choices, guaranteed fast response times and optimization functions. The method supports an easy maintenance process with a unique consistency check of configuration data. We compare the tool with previously proposed methods like expert systems to show the advantages of this approach. An example of car configuration is presented.

Key-words.

Product-oriented information systems, AI, Design for Manufacturability/ Dependability, Testability, Product Configuration, Constraints.

1 Introduction.

Products generally become more complex with more variants and shorter life cycles. Wrongly configured products means increased expenses, unsatisfied customers and stressed sales people therefore the need for good sales tools to assist the configuration process is evident.

salesPLUS [2] exploits the recent development in constraint satisfaction algorithms to evolve a new methodology for interactive configuration.

2 Methodology .

2.1 Objects & Resources.

The most fundamental part of the configuration data is the object. All the items or parts that can go into a product must be represented as objects. Product features may also be represented as objects e.g., the screen resolution on a PC although it derives from the combination of Monitor and Videocard.

The objects have a number of attributes.

Object :: {Type, Id, Resources, Menu position, Description, Default-value,...}

A resource is a property or measure of some objects. Examples of a resource are "Price", "Memory", "Weight", etc. Objects can have positive or negative resource values. Hence it is possible to have a provider/consumer principle, e.g., devices use certain amounts of power whereas power-supply's provide more power.

2.2 Constraints.

The representation of a configuration problem is naturally done with constraints governing the objects. These constraints may stem from physical limits (as space limits) and from technical restrictions (as "supports only") and from marketing demands (only sell some combinations). This is the contents of the configuration problem. This product information can typically be found in a printed configuration guide and transformed on a one to one basis into constraints.

Two types of constraints are supported: boolean algebra and finite domain arithmetic. Examples from a UNIX workstation.

```
Storage >= 300
  "The system should entail at least 300 Mb of storage (resource)."

RAM_8Mb + RAM_32Mb + RAM_64Mb <= 2
  "The model can only accommodate two optional memory options."

impossible( Model_A,   CDROM )
        "Model A does not support CDROM."
```

The representation of UNIX workstations consists of only 30-40 constraints.

3 Consistency check.

The constraints only state facts and there is no meta reasoning dealing with which rules to use. Since each rule represents a real world constraint they can mentally be checked individually.

salesPLUS has a consistency checker that can detect inconsistencies in individual constraints, bound objects and most important inconsistency in the overall configuration data.

Explanation and traces of conclusions can also be made for testing purposes.

4 Deduction methods of inference engine.

4.1 Constraints.

With traditional programming as with production rules systems the programmer has to master the general information flow. As a result, these systems are very intricate and they are hard to develop, test and maintain when either incomplete information or unordered information flow has to be dealt with. In contrast to constraint programming where you deal with problem contents only.

The basic step in the configuration process is the following: Given the partial configuration selected so far, the existence of a corresponding valid configuration has to be verified (consistency), and the elements that can still be selected have to be known (domain reduction). There is both unordered information flow and incomplete information available in a typical configuration problem.

The inference engine in salesPLUS is a State of the Art constraint solver [1]. It handles both boolean algebra and finite domain arithmetic constraints based upon interval calculus and the patented Beologic array inference technology.

Having a pure constraint representation allows the user to add additional constraints at runtime like "The computer should have more than 32 Megabytes of memory", and salesPLUS uses this information to reduce the set of possible machines.

Having asserted the additional constraints it is now possible to minimize the price resource with an optimization algorithm. The resulting computer will be the cheapest one that have more than 32 Mbyte of memory. This strong feature is impossible in traditional configuration systems.

4.2 Time Complexity.

The inference engine exhibits a nice and polite policy towards the user. It has guaranteed response time that can be user selected independent of the platform.

The inference engine's job is to check that no inconsistency occurs between the constraints and the assignments. Determining solvability to a set of objects that must satisfy a set of constraints is NP-complete i.e. the execution time grows exponentially with the size of the problem.

The inference engine may therefore use longer time than desired to establish all consequences. The engine will then give control to the user when the specified time (1-2 seconds) is used or sooner.

What the inference engine can deduce, increases with the number of assignments of objects. This is so because the search space only depends on the number of unbound objects.

For "large" products this means that an option may seem to be free, but when you select it the inference engine will find that it is not possible and assert this. In practise this is not a problem with the products that have been modelled.

The concept of the inference engine allows it to take advantage of future hardware improvements, and it will not be slow on weak computers.

5 Comparison with traditional configuration Systems.

5.1 Representation.

Configuration problems are by nature NP-complete. Traditional configuration systems deal with this problem in several ways.

* For expert system rules that deals with the execution of the application are added to reduce the problem to a simple linear/polynomial problem. Adding these rules is very bad for maintainability because they are not founded in the problem contents but only in the context.

% salesPLUS does not have rules that deal with execution issues.

* For decision trees the user is forced to make his choices in a predefined order. This reduces the problem by avoiding unordered information flow. This yields an application that is very constraining for the user. Imagine that a customer wants a red car. The system will prompt for model,engine,.. etc. And when it comes to the colour it might not be possible to select red at all.

% salesPLUS gives full freedom in the order of choices.

* Batch mode validation does not allow for interactive configuration therefore the problem is reduced by avoiding incomplete information. Batch mode means that a user picks all the parts that he assumes appropriate. Then the validation algorithm produces a list of missing and conflicting parts. Here the user has to put up with an iterative process. Imagine that the customer specifically said that he wanted a part and the configurator then says that it is not possible. Then he has to start all over again.

% salesPLUS has an interactive mode where the status of resources are displayed dynamically such as price, storage and weight to give the user better overview.

5.2 Maintenance.

Maintenance of the product model is an extremely important feature of a configuration system where product changes may occur on a monthly basis or even more frequent.

Traditional configuration tools have difficulties with rapid changing product models. For expert systems it is known that rule bases with up to 10000 rules exists [5] which unfortunately makes it hard for even experienced engineers to change the rules.

Easy maintenance in salesPLUS is achieved by keeping the number of constraints small. A single constraint often replaces many production rules and a company's products should be split into several knowledge bases. It is therefore possible for even quite complicated products to be represented by less than 100 constraints.

Before release of any configuration system a test must be made to see if errors occur in the configuration data. With the product range decomposed it is only necessary to test the changed products. Whereas if you have one big knowledge base, you might have introduced side effects and the whole product range must be tested again.

Concurrent product configuration data maintenance is possible with salesPLUS.

6 Discussion.

The reason for the progress made in salesPLUS is the recent development in constraint

satisfaction algorithms [3][4] that allows them to successfully solve real world problems.

The representation in salesPLUS is restricted to constraints on boolean and finite domain. It does not allow expressions like "exp(cos(x))". However most configuration tasks that sales people are able to perform with the printed documentation and sufficient time is suitable for salesPLUS. With salesPLUS it has been established that you can make configurations for cars, trucks, PABX's, computers, doors, coffee automates, custom installation guides, water pumps, hydraulic proportional valves, airplanes etc.

7 Example: Car Configuration.

The configuration problem is from SAAB Automobiles 1992 with the models from the 900 series. We will show how products can be modelled within the salesPLUS representation.

In the example only the objects and constraints will be considered in depth except for the Object "Models". Therefore issues related to menu positioning, ID, Button names, Descriptions, object ID's and Languages are omitted. A graphics tool, salesPLUS Definer exists that helps the user to declare objects, resource, menus etc in an easy way and enables users to make rules by pick and place. This tool will not be described here.

7.1 Declarations.

First the objects are declared.

The 3 models are SAAB 900, Cabriolet and Turbo Cabriolet.
They are declared as an object "Models" of the type ONEOF.

```
Object Type: ONEOF
        Name: Models
        Identifier: Model
        Description: "Some of the models in ....."
        Parent Menu: TOPLEVEL
        Position X,Y: 1,1
        Default Value: SAAB_900
        Elements: Name: "SAAB 900"
                        Identifer: SAAB_900
                        ID/Part No.: xxx-yyy
                        Description: "The most safe car ...."
                        Resources: Price=file , Weight=1112
                  Name: "Cabriolet"
                        Identifier:Cabriolet
                        ID/Part No.: xxx-qqq
                        Description: "Unique features ...."
                        Resources: Price=file , Weight=1123
                  Name: "Turbo Cabriolet"
                        Identifier: Turbo_Cabriolet
                        ID/Part No.: xxx-zzz
                        Description: "The ultimate driving   ...."
                        Resources: Price=file, Weight=1234
```

The 4 engines are 2.0i, 2.1i, 2.0S and Turbo.
They are declared as one object "Engine" of the type ONEOF

Available accessories are Automatic gearbox, ABS-brakes, Air Bag, Air-condition, Audio system, Automatic air-conditioning, Eec. mirrors/windows and Cruise control. These are declared as eight objects of the type SINGLE.

Available Sunshine roofs are Manual steel, Electric steel and Electric glass.
The sunroofs are declared as one object "Sunroof" of the type ATMOSTONE.

Available Trims are Velour Jet-Tuff Horizon, Velour pique Parallel, Leather Contour and Leather Suede Contour. The trims are declared as one object "Trim" of the type ONEOF.

Available trim Colors are Labrador, Marine, Puma, Angora, Buffalo and Pamir.
These are declared as one object "TrimColor" of the type ONEOF.

Available standard paints are Cirrus white, Black, Embassy blue, Cherry red and Talladega red. The paints are declared as one object "Stdpaint" of the type ATMOSTONE.

Available metallic paints are Citrin beige, Platana grey, Le Mans blue, Scarabe green and Monte Carlo yellow. Declared as one object "Metalpaint" of the type ATMOSTONE.

Object that do not relate to physical items is Leather and delivery time. Leather is declared as an objects of the type SINGLE. Delivery time is declared as one object of the type ENUM [0-35].

Declared resources are Price, Weight and Horse power. Declared menus are Accessories, Trims, Paints, Accessories at Dealer.

7.2 Constraints.

SAAB configuration guidelines:

*A It is impossible to have both Air conditioning and automatic Air conditioning.
*B The turbo cabriolet comes with Turbo engine, metallic paint, leather trim and cruise control.
*C Ordinary cabriolets comes with 2.1 litre engine.
*D Ordinary cabriolets cannot have the paint Marine.
*E Models with red, grey or green paints cannot be ordered with Marine trim.
*F Models with beige or green paints cannot be ordered with Puma trim.
*G Cars with Turbo engine should be ordered with ABS brakes.
*H There cannot be ordered sunroof for cabriolets.
*I Delivery times are 14 days for the SAAB 900, 21 days for the Cabriolet and 35 days for the Turbo Cabriolet.

These configuration guide lines yields the following knowledge base. Note the almost one to one correspondence between the written guidelines and the constraints. In the constraints the objects identifiers are used.

Knowledge base:

NEW: Leather <> Trim[Leather_Contour] or Trim[Leather_Suede];
"Sub grouping of leather trim types"

NEW: Leather <> TrimColor[Pamir] or TrimColor[Buffalo];
"Sub grouping of leather trim color"

A : Impossible(AirCondition, Automatic_AirCondition);

B: Model[Turbo_Cabriolet] -> Engine[Turbo] and
Metalpaint and Leather and cruise_control;

C&D: Model[Cabriolet] -> Engine[s21] and not TrimColor[Marine];

E: Stdpaint[Cherry_red] or Stdpaint[Talladega_red] or
Metalpaint[Citrin_beige] or Metalpaint[Scarabe_green] -> not TrimColor[Marine];

F : Metalpaint[Citrin_beige] or Metalpaint[Platana_grey] -> not TrimColor[Puma];

G: WARNING (Engine[Turbo] and not ABS);

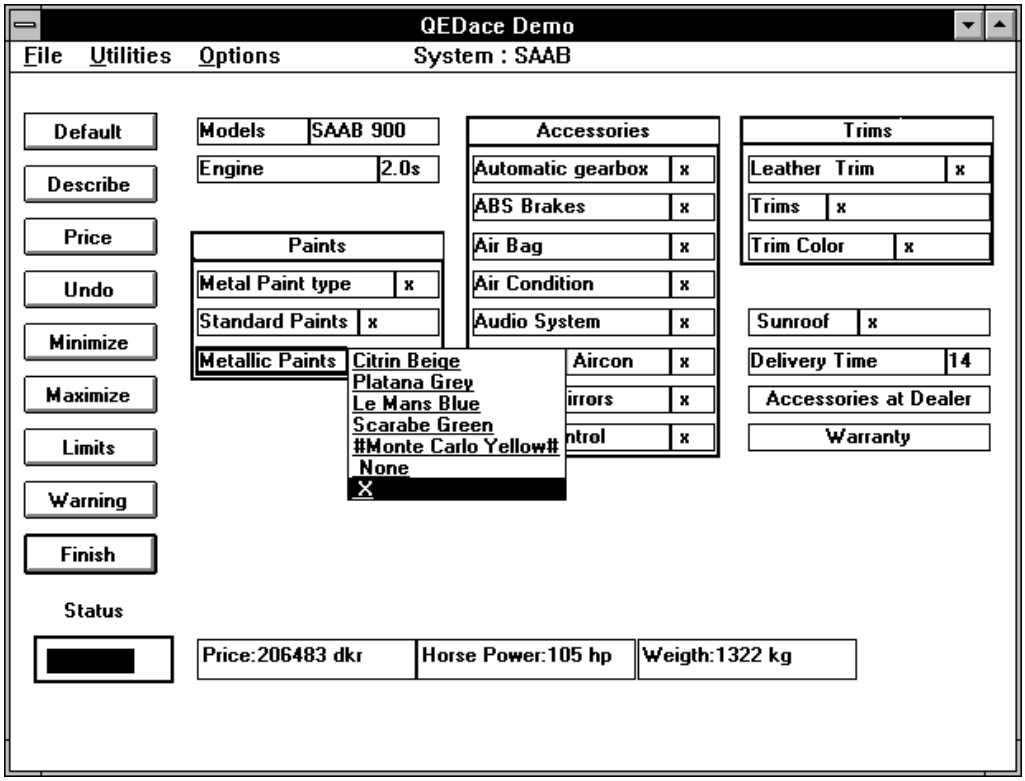
H: Model[Cabriolet] or Model[Turbo_Cabriolet] -> not Sunroof;

NEW: Metalpaint or Stdpaint;
"You must pick one type of paint"

I: Delivery_time == 14 * Model[SAAB_900] + 21 * Model[Cabriolet]
+ 35 * Model[Turbo_Cabriolet];

A warning constraint is a constraint that can be overruled but for most cases this is not desirably. For cars the boolean constraints are used mostly but for other products like PABX's the configuration model consists mainly of arithmetic constraints because most options can come in different numbers.

7.3 Runtime Sales Tool.



With these declarations and constraints the end user sales tool is shown in fig. 1. The leftmost column of buttons is standard functions that salesPLUS offers the user. The resources are dynamically displayed at the bottom of the window. The objects are shown above the resources. Two selections have already been made in the window namely Models = "SAAB 900" and Engine = "2.0s" .The box menu "Paints" is used to group logically related objects. The sub menu "Accessories at dealer" is shown when activated.

With the objects you can configure in the naive way by picking as desired. In the opened list for "Metallic Paints" you will notice that the option "Monte Carlo Yellow" is surrounded by # which means that this option is not available any more due to the constraints. However if you insist on this paint; select it and the system will resolve which earlier selections that have to be overruled.

The runtime system of salesPLUS holds a number features that is not described in any length here like Trace, Total list, Discounts, Finish, Default, Response times, Load/save resource, Load/Save configuration etc.

8 Implementation.

In salesPLUS a **Definer** environment exists that runs on MS-Windows, and a executable runtime **Customizer** environment that runs on MS-DOS, MS-Windows and MOTIF. The runtime interface is build upon an ANSI-C API set, that can easily be ported to any platform. SalesPLUS does not demand any proprietary application such as database etc. The inference engine uses a compiled version of the constraints for efficient execution.

References.

[1] CLP(B). Logic Programming with Boolean Constraints. Final report.
Henri Beringer, Centre Scientifique IBM France. Dec. 1991.

[2] Concept of the BEOLOGIC Product Configurator. Beologic A/S, 1994.

[3] Constraint satisfaction in Logic Programming. Pascal Van Hentenryck. The MIT Press. 1989.

[4] Algorithms for Constraint-Satisfaction Problems: A Survey. Vipin Kumar. AI Magazine. Spring 1992.

[5] Expert System for Configuration at DIGITAL: XCON and beyond. Communication ACM (vol. 31-32) Dec. 1988.