A Configuration Tool to Increase Product Competitiveness

Bei Yu and Hans Jørgen Skovgaard, Baan Front Office Systems

THE SALESPLUS PRODUCT-CONFIGURATION TOOL EFFECTIVELY SOLVES COMPLEX CONFIGURATION PROBLEMS, REDUCING COSTS WHILE MEETING CUSTOMER EXPECTATIONS IN REAL-WORLD APPLICATIONS.

MANY COMPANIES NOWADAYS must rapidly develop and deliver products and product variants in a short time to meet increased customer demands and stiffer product competition. Product configuration (determining a set of components and their relationships, and composing them into a product that satisfies customer requirements and design constraints) therefore faces the problem of quickly providing these variants without incurring high costs.

However, modern products have become increasingly complicated, incorporating many different technologies. Incorrect configuration solutions will inevitably require corrections and lead to high costs. So, product configuration also faces the challenge of delivering a solution that is right the first time.

Current configuration tools have not met these challenges, for three main reasons:

- They can't handle product complexity well enough. Complicated product structures with numerous relationships lead to the hard configuration problems. No real solutions exist yet for these problems—that is, no approach to product configuration can significantly decrease product complexity.
- They can't support configuration maintenance well enough. Although several theoretical approaches are available, they are impractical, because of the limitations of time, computer speed, memory, and so on.
- They can't maintain product knowledge well enough. Some maintenance techniques in existing configuration tools are also expensive and time-consuming.

The salesPLUS[1] tool can handle the complexities of product configuration, with highly efficient inference-engine performance and application maintainability. This commercial AI-based system effectively and intuitively models and configures products. It effectively produces and maintains consistent, accurate configurations that meet customer demands while significantly reducing costs.

The salesPLUS system

salesPLUS is based on the concept of mass customization—that is, product configuration generates customized solutions based on a standard product or product model.[2] It adopts the computer-support-assistant philosophy: it is an assistant interacting with the user.[3] The goal of using an AI-based approach was not superior performance, because the algorithmic system starts out from a precompiled representation and is, as expected, several times faster, but maintenance savings.[4] The main objectives of salesPLUS are to

- ensure configuration correctness—that is, 100% accuracy on the valid configuration solutions;
- develop an effective approach for handling configuration consistency;
- handle configuration constraints;
- overcome the limitations of product-knowledge maintainability; and
- support the whole development of a configuration application—that is, product modeling and configuration—without requiring programming skill.

1094-7167/98/$10.00 © 1998 IEEE

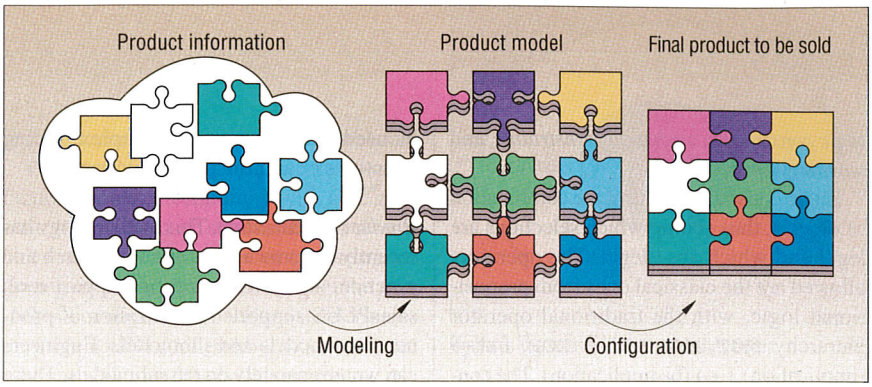

Figure 1. The configuration-design process.

For the whole life cycle of product development, configuration consists of two aspects: *configuration design* and *configuration maintenance*.[5] Configuration design deals with creating configuration solutions; it involves selecting elements and the ways of configuring them. Configuration maintenance deals with maintaining a consistent configuration under change; this involves the consistency among the selected elements and decisions. When a decision for selected elements changes, configuration maintenance must trace all the decisions that are related to the changed decision and revise them, if necessary, to maintain consistency among the elements and decisions.

salesPLUS divides configuration design into two stages: *product modeling* and *configuration* (see Figure 1). The product-modeling stage defines the product model and the classification (assortment) of products and parts by means of the objects, properties, allowable property domain, and constraints. The configuration stage creates a specific product instance or variant based on that model. This configuration solution can be represented as a product specification, a sales order, or a parts list or bill of materials.

salesPLUS incorporates configuration maintenance into the product-modeling and configuration stages. For configuration maintenance, the systems uses a constraint-based approach that involves fewer, more intuitive constraints. It performs problem-solving to ensure that configuration is consistent and correct.

Figure 2 gives a scenario for product configuration in salesPLUS. First, based on product information, salesPLUS creates product models. Then, through computer compilation, the system prepares the product models for configuration. Next, salesPLUS helps the end user, who might be a designer, a sales engineer, or even a customer, make decisions on configuration details. Finally, it generates the configuration solutions.

Product modeling. A product model incorporates all the information that represents products or services. This information is encapsulated in *objects*, which involve resources and constraints. A model can consist of several submodels; for example, a train consists of several cars. Objects might vary from physical parts (such as a screw), to subassemblies (for example, a speaker), to whole products (perhaps a car radio system).

Types of objects. As part of the selection process during configuration, an object might involve an option that is yes or no, is a fixed number, or is an interval. Or, it might have a list of options where none, one, at least one, or several of the options must be picked.

Accordingly, objects fall into these types:

- `Single` objects have no fixed relations or multiple instances. These objects have a selection option of yes or no.
- `Enum` objects can be enumerated and stated in a range—for example, [0..32].
- `Interval` objects let you can divide a numeric range into a number of consecutive nonoverlapping subintervals. With these objects, you can reason about the interval, not the exact value.
- `OneOf` objects have mutually exclusive elements. They are used to group a set of components from which one must be selected.
- `AtMostOne` objects let you select one element at most. Otherwise, they are similar to `OneOf` objects.
- `AnyOf` objects let you select no elements or several elements at a time. They are typically used when you have a set of options that can be selected individually. `AnyOf` objects are usually used in rather compact presentations or for a logical grouping of items in the product model.
- `Time` objects resemble `Single` objects. The only difference is that they are `FALSE` until a specified activation time, after which time they obtain the value `TRUE`. You can use `Time` objects, for example, to compensate for the asynchronous aspect of product events and product model releases.
- `Info` objects support textual information. The text passes information on the product or some components to the user.

Constraints. In salesPLUS, the relationships between objects in the product model, as well as design requirements, are constraints. The constraints deal with only the contents of the product model problem, not the computational aspects. Thus, they deal only with stating the problem, not programming the solution. There are three kinds of constraints:

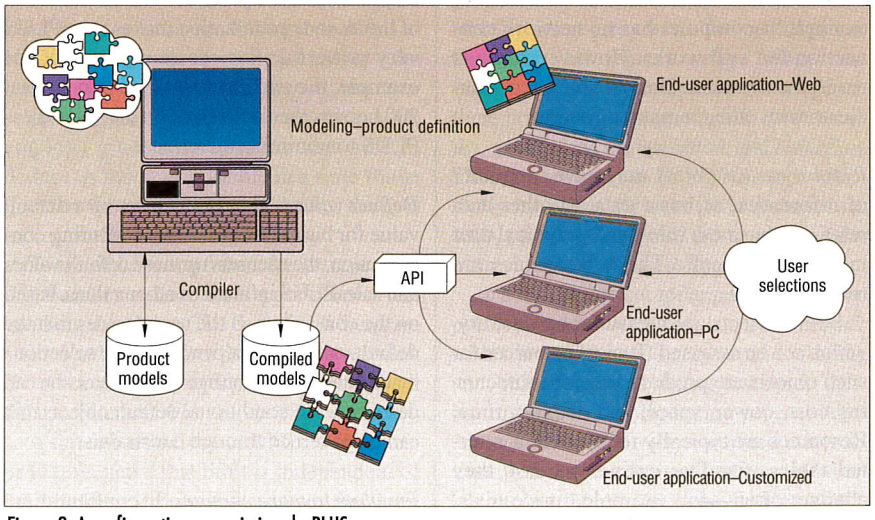

Figure 2. A configuration scenario in salesPLUS.

logic constraints, *arithmetic constraints*, and *warning constraints*.

Logic constraints control the combination of objects, thus stating which selections are legal and which are illegal. The operators allowed are the classical ones from propositional logic, with the traditional operator hierarchy: **NOT**, **AND**, **OR** | **XOR**, and → (implication) | ↔ (bi-implication). The constants **TRUE** and **FALSE** can also be used. The system also allows many special operators having a comma-separated list of variables as arguments. Such operators are shorthand for complicated logic expressions:

- **Allequal** (The elements must be either all **TRUE** or all **FALSE**.)
- **OneOf** (There must be exactly one **TRUE** element.)
- **AtMostOne** (There must at most be one **TRUE** element.)
- **AtLeastOne** (There must at least be one **TRUE** element.)
- **Impossible** (The elements cannot all be **TRUE**.)

Arithmetic constraints set physical (numerical) limits—for example, for determining the power supply or the length of a train car. The operators allowed are the classical ones from mathematics, + (addition), – (subtraction), and * (multiplication); and one of these comparison operators: == (equal [compare]), >= (greater than or equal to), <= (less than or equal to), > (greater than), or < (less than).

Warning constraints can be violated without being fatal to the product configuration. Their violation means that you are in a very special situation and must be careful. For example, a computer has no network connection but still works. However, in most cases, the computer must be able to communicate with other computer systems.

Resources. An object may have a number of references, as basic data, to other data relations, business rules, and graphical user interface elements. These references are resources.

Some objects are measured by quantity, which can be assessed. Typical resources for such objects are product price, manufacturing cost, power, space, and delivery time. Resources are typically referenced in external tables, fixed or online, because they change often—for example, price lists, which vary per country. Database links can be used as references for complex pricing structures or additional information.

Product modulization. This methodology has recently become a hot topic in research and practice. As a configuration-support tool, salesPLUS supports modulization of products into models and submodels. Engineers can work separately on the submodels. These submodels can then be linked into a whole product model. This is important for an integrated development environment; modulizing a large product into manageable modules can reduce product complexity.

The configuration process. During configuration, salesPLUS makes a series of selections based on the customer and design requirements, and maintains consistency between the selected elements and decisions. Some significant features of this process are

Order-free selection. Unlike a decision-tree-based configuration, configuration decision making in salesPLUS is order-free. In other words, the user can start from any selection, in any order. This feature gives the user a lot of freedom to carry out configuration, compared to decision-tree or specific selection-dependency rules.

Limits. Resources can be given upper and lower limits during configuration. This feature lets the end user focus on needs (behaviors and performance), rather than on selecting objects.

Optimization. salesPLUS has built-in optimization facilities. Both minimization and maximization are possible. The combination of limits and optimization makes salesPLUS very powerful. For car configuration, for example, the end user can state that at least 200 horsepower is needed and then ask salesPLUS to minimize the price.

Default values. The user can set up a default value for his or her own reasons. During configuration, the user sets up these default values and salesPLUS inferences other values, based on the constraints. If the user decides that the default values will depend on other selections made later during configuration, he or she can define a set of conditional default objects that can be governed through constraints.

Freedom to make changes. In configuration, changes of previous choices are typical and inevitable. In salesPLUS, a user can change his or her decisions without worrying about the configuration's consistency.

System architecture. salesPLUS consists of four primary modules (see Figure 3).

The *Definer* generates a product model that declares the relationships between elements and constraints. With the Definer, modeling engineers model a product by means of a set of objects with constraints. It has facilities for checking the model's consistency, for simulating and running the end-user application, and for creating the interface for carrying out configuration.

The *Customizer* carries out configuration based on the product model. This process is interactive, to satisfy specific customer needs and requirements. The result of the user's selections and the system's subsequent conclusions is a specific configuration solution. Solutions can be printed, saved, or exported to other product-development tools.

The *Kernel* holds an internal representation of the product models and contains the heart of the system: the inference engine, which ensures configuration consistency and correctness. We'll discuss the engine in more detail in the next section.

The *salesPLUS API* (application programming interface) links application-interface modules such as the Definer and Customizer with the Kernel. Also, by accessing a number of kernel functions through the API, users can easily add new user-interface features to the Customizer or even develop new user interfaces.

The user-friendly interface at the configuration stage is a purely data-driven interface based on product models. That is, when product models change, salesPLUS automatically creates an updated user interface based on the changed models. In addition, salesPLUS can support configuration over the Internet, using the same model created through the Definer.

A product-configuration tool should be able to build a feature-rich product, should be based on an open architecture, and should easily integrate with other enterprise applications. We've achieved this in salesPLUS through product modulization and the incorporation of well-defined integration points. This minimizes the effort required to implement the product in almost any environment. For example, we've developed an add-in interface to support customers' special needs.

Inference engine design. The inference engine's objective is to handle constraints such that correct configuration solutions can be derived as quickly as possible. It is an effective constraint-based problem solver with our patented technology[6] that enables the entire system to carry out complicated product configuration in an effective time.

During configuration, avoiding situations where selections contradict each other is very important. Some configuration systems have addressed this inconsistency issue, using different approaches. Our inference engine uniquely ensures such consistency early in the decision-making stage—that is, it can reason with incomplete information within a guaranteed response time.

For example, assume that there are two constraints, "$A \rightarrow B$ or C" and "B or $C \rightarrow D$." If we know that A is true, we can directly deduce D to be true without knowing the values of B and C. In many configuration systems, however, D cannot be inferred directly through the two separated constraints. This situation might cause inconsistency because it lets the user assign D. If, for example, we assign D to be false, the next evaluation of B or C will cause a conflict.

Many systems handle this problem by informing the user that a contradiction exists and displaying the conflicted constraints. The user must handle the problem, either by using backtracking or by adding another constraint, "$A \rightarrow D$," to avoid the conflict. Nevertheless, applying backtracking will result in time-consuming configuration, and adding a new constraint will lead to redundant constraints that are not directly linked to the product, thus complicating maintenance.

The salesPLUS inference engine, in contrast, can avoid poor performance and redundant-constraints maintenance while ensuring proper configuration. To provide configuration consistency and correctness, it employs *complete deduction*, *graceful degradation*, *dynamic consistency maintenance*, and *Boolean constraint handling*.

Complete deduction. A configuration problem is NP-hard; that is, it grows exponentially in time with the complexity of the product to be configured. The number of product elements, the various constraints, and the product model's level of detail cause this complexity. This kind of problem leads to a large search space for solutions. Although some algorithms such as forward-checking perform better than traditional backtracking approaches, time efficiency is still a problem.[7]

Complete deduction uses constraint-satisfaction-problem technology with time efficiency. This approach starts with a set of variables with domains and a set of constraints among these variables. Then, when the user makes a configuration decision, it determines the current valid configuration solution space without violating the given constraints.

Complete deduction consists of a series of algorithms with increasing time complexity. These algorithms work together to carry out three basic sequential steps:

(1) Propagation—making all the relevant constraints and examining their consequences, based on the configuration.
(2) Determination of solvability—evaluating each undetermined selection option to see whether a corresponding configuration solution exists.
(3) Completion—guaranteeing that all possible consequences have been determined.

This approach, in a sense, maintains configuration consistency by ensuring a valid configuration solution space in the runtime environment. In other words, selections are bound, in a shrinking configuration solution space, toward solutions while configuration progresses. This approach guarantees the correctness of solutions by pruning inconsistent selection choices from the selection space.

The number of the basic steps that complete deduction carries out depends on three factors: the specified maximum allowed time (see the next section), the product's complexity, and the computer's speed.

Graceful degradation. When a problem is NP-hard, the inference engine might take longer than desired to establish all the consequences. So, to assist the user, the engine employs graceful degradation: it tries to deduce as much as possible in a time frame given by the user. This method gives the user more control over configuration.

To specify the time frame, the user specifies the `MaxTime` time-limit parameter when invoking the engine. The engine will then give control to the user when that time limit expires. It can then run as a background process until the user interacts with the system.

A suitable time limit should be around one or two seconds; this limit is independent of the platform. In our experience, the more selections the user makes, the less time the inference engine needs to deduce the conclusions, because the configuration solution space shrinks during configuration.

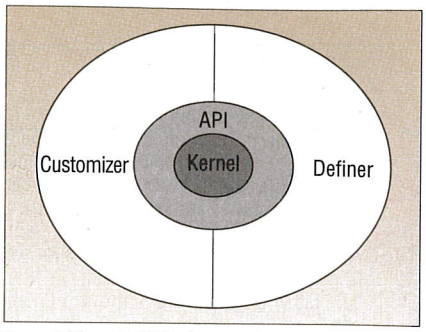

Figure 3. The salesPLUS system architecture.

Dynamic consistency maintenance. If the user tries to change a selection, and this change will negatively affect the configuration's consistency, the system gives a warning. If the user continues with the change, the system provides a list of selections that have been made that relate to the selection to be changed. This list informs the user that these selections will be affected and must be reselected to meet the change requirements. The user can decide which alternatives from the selection list must be reselected. Once the user decides to redo selections, the system eases the relevant bound variables in the constraints.

Boolean constraint handling. The inference engine's power comes from the unique Boolean constraint-handling mechanism—that is, *binary array-based logic mathematics*.[8] This mechanism has been successfully applied to propositional logic in a finite domain.

During compilation, salesPLUS compiles the product model with constraints to create several truth tables. In array-based logic, any propositional form is transformed into a truth table in binary-array form. The mathematical properties of logical arrays allow the execution of any logical inference through just three primitive array operations: *outer product, generalized transposition*, and *reduction*. These operations can be implemented effectively by using the com-puter instructions `AND`, `OR`, and `XOR` on 32 bit-words, thereby achieving parallel processing.

In addition, the compiler preprocesses the product model such that it contains truth tables in compressed-array form, including attached information such as resources. During runtime the configuration process takes advantage of the already compiled information. Referencing the truth table at runtime lets the configuration engine deduce all the consequences faster than would an engine

A: It is impossible to have both air-conditioning and automatic air-conditioning.
B: The Turbo Cabriolet comes with the Turbo engine, metallic paint, leather trim, and cruise control.
C: An ordinary Cabriolet comes with the 2.1-liter engine.
D: Ordinary Cabriolets cannot have the trim color 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 the Turbo engine must be ordered with ABS brakes.
H: A sunroof cannot be ordered for Cabriolets.
I: The delivery times are 14 days for the Saab 900, 21 days for the Cabriolet, and 35 days for the Turbo Cabriolet.
J: Either air bag, or ABS brakes, or both should be selected.

(a)

```
NEW: Leather_Trim ↔ Trims[Leather_Contour] OR
     Trims[Leather_Suede]; "Subgrouping of leather trim
     types"
NEW: Leather_Trim ↔ TrimColor[Pamir] OR
     TrimColor[Buffalo]; "Subgrouping of leather trim
     color"
A:   IMPOSSIBLE(AirCondition, Auto_AirCon);
B:   Models[Turbo_Cabriolet → Engine[turbo] AND
     MetallicPaints AND Leather_Trim AND Cruise_Control;
C&D: Models[Cabriolet] → Engine[e21i] AND NOT
     TrimColor[Marine];
E:   StandardPaints[Cherry_Red] OR
     StandardPaints[Tallaga_Red] OR
     MetallicPaints[Citrin_Beige] OR
     MetallicPaints[Scarabe_green] → NOT
     TrimColor[Marine];
F:   MetallicPaints[Citrin_Beige] OR
     MetallicPaints[Platana_Grey] → NOT TrimColor[Puma];
G:   Engine[turbo] → ABS_Brakes;
H:   Models[Cabriolet] OR Models[Turbo_Cabriolet] → NOT
     Sunroof;
NEW: MetallicPaints OR StandardPaints; "You must pick one
     type of paint"
I:   Delivery_Time == 14 * Models[Saab_900] + 21 *
     Models[Cabriolet] + 35 * Models[Turbo_Cabriolet];
J:   WARNING (NOT Air_Bag AND NOT ABS_Brakes);
```

(b)

Figure 4. Saab configuration constraints at the (a) application level and (b) system level.

that merely goes through a list of programming steps.

Car configuration with salesPLUS

Now let's look at a specific application of salesPLUS. The configuration problem is from the 900 series models of 1992 Saab automobiles. We'll consider the objects and constraints in depth. We've omitted some issues related to the Definer's graphical interface, such as menu positioning, ID, button names, descriptions, object IDs, and languages.

Model object and resource declarations. First, modeling engineers declare objects through the Definer.

- The three models are the Saab 900, Cabriolet, and Turbo Cabriolet. They are declared as an object `Models` of the type `OneOf`.
- The four engines are the 2.0i, 2.1i, 2.0S, and Turbo. They are declared as one object `Engine` of the type `OneOf`.
- The available accessories are automatic gearbox, ABS brakes, air bag, air-conditioning, audio system, automatic air-conditioning, electric mirrors and windows, and cruise control. These are declared as eight objects of the type `Single`.
- The available sunroofs are manual steel, electric steel, and electric glass. They are declared as one object `Sunroof` of the type `AtMostOne`.
- The available trims are velour jet-tuff horizon, velour pique parallel, leather contour, and leather suede contour. They are declared as one object `Trims` of the type `OneOf`.
- The available trim colors are labrador, marine, puma, angora, buffalo, and pamir. They are declared as one object `TrimColor` of the type `OneOf`.
- The available standard paints are cirrus white, black, embassy blue, cherry red, and talladega red. They are declared as one object `StandardPaints` of the type `AtMostOne`.
- The available metallic paints are citrin beige, platana grey, le mans blue, scarabe green, and monte carlo yellow. They are declared as one object `MetallicPaints` of the type `AtMostOne`.
- The object `Leather_Trim` groups the objects that are related to leather features such as trim and trim color. It is declared as an object of the type `Single`.
- An object that does not relate to physical items is `Delivery_Time`. It is declared as an object of the type `Enum`.

Several resources are declared in a database form. The declared resources are `Price`, `Weight`, and `Horsepower`. The declared menus are `Accessories`, `Trims`, `Paints`, and `Accessories at Dealer`.

Configuration constraints. Figure 4a shows some application-level constraints for Saab configuration. The application-level constraints can be interpreted into the system-level constraints. Figure 4b shows some system-level constraints.

Note the almost one-to-one correspondence between the written guidelines (the application-level constraints) and the system-level constraints. These constraints use the object identifiers. Only a few new extra constraints have been added to represent the whole meaning of the constraints at the application level. The system-level constraints are intuitive and very readable.

Figure 5 shows the Saab objects and constraints using the Definer.

Saab configuration. Figure 6 shows the status of the Saab configuration process through the Customizer. The user can decide the layout of the selection interface when creating the product model with the Definer. The system makes deductions for selection based on the constraints, corresponding to the user decisions. Selections can be made by the user (the light-green checks in Figure 6) or by the system (the dark-green checks). The resources are calculated and updated simultaneously while decisions on selections are being made. System functions such as *reset, default, undo,* and *finish* let the user carry out

configuration without worrying about configuration correctness and consistency.

BASED ON OUR ACHIEVEMENTS with salesPLUS (see the sidebar for two other successful applications), we plan to continue our research into computer-supported product development. We'll enhance the problem-solving ability and system functionality of salesPLUS, to cope with various configuration problems throughout product development. Our product will be a solution that can cover a wider range of applications, spanning from front-office business to back-end engineering. Our goal is to make the configuration system an integrated solution in a common product-development environment. Toward that end, our next step will be to emphasize system and data integration, so that product information and solutions can be shared, exchanged, and applied among different tools in the individual product-development stages. ■

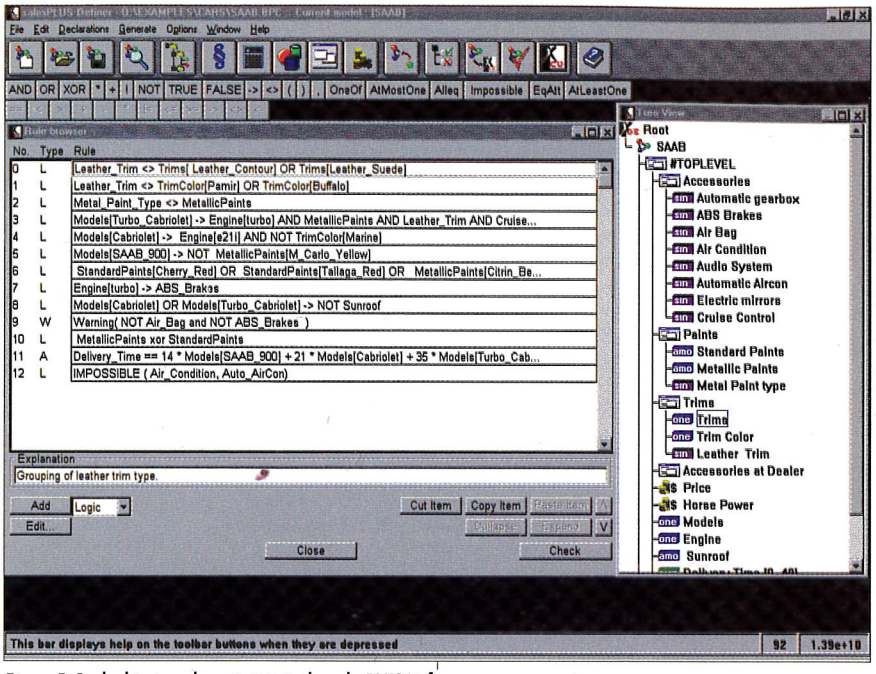

Figure 5. Saab objects and constraints in the salesPLUS Definer.

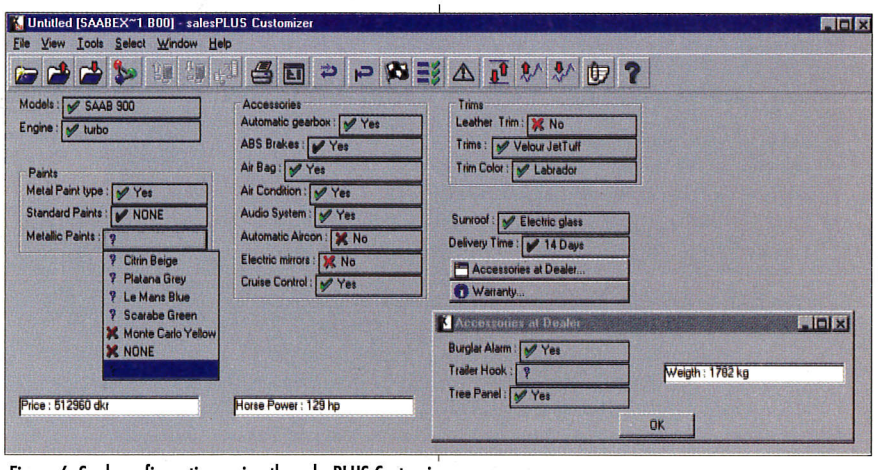

Figure 6. Saab configuration, using the salesPLUS Customizer.

References

1. H.J. Skovgaard, "A New Approach to Product Configuration," *Proc. ilce '95: Third Int'l Conf. Integrated Logistics & Concurrent Eng.*, 1995, pp. 197–204.

2. B. Victor and A. Boynton, *Invented Here: Maximizing Your Organization's Internal Growth and Profitability*, Harvard Business School Press, Boston, 1998.

3. K.J. MacCallum, "Does Intelligent CAD Exist?" *Artificial Intelligence in Eng.*, Vol. 5, No. 2, Apr. 1990, pp. 55–64.

4. C.J. Thornton and B. du Boulay, *Artificial Intelligence through Search*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

5. B. Yu, "A Virtual Configuration Workbench for Product Development," PhD Thesis, Univ. of Strathclyde, Dept. of Design, Manufacture, and Eng. Management, Glasgow, Scotland, 1996.

6. G. Møller, "Signal Processing Apparatus and Method," Patent WO 90/9001, 1989.

7. M. Shanahan and R. Southwick, *Search, Inference and Dependencies in Artificial Intelligence*, Ellis Horwood Ltd., Chichester, UK, 1988.

8. G. Møller, "On the Technology of Array-based Logic," PhD Thesis, Technical Univ. of Denmark, Dept. of Electric Power Eng., Lyngby, Denmark, 1995.

Bei Yu is a research engineer at the Configuration Competence Centre, Baan Front Office Systems. She has been doing research in product configuration since 1992. Her research interests include product modeling, the configuration process, configuration technology, and artificial intelligence. She received her BSc in computer science and PhD in product configuration from the University of Strathclyde, Scotland. Contact her at Baan Front Office Systems, Hørkær 12A, DK-2730 Herlev, Copenhagen, Denmark; byu@baan.com.

Hans Jørgen Skovgaard is the vice president for configuration development at Baan Front Office System A/S and is the chief architect on the configuration product salesPLUS, with whose development he has been involved since 1990. His research interests include black-box constraint solvers and declarative language design. He received his BSc in electronic engineering from Odense Polytechnic, Denmark, and his MSc in artificial intelligence from University of Edinburgh, Scotland. Contact him at Baan Front Office Systems, Hørkær 12A, DK-2730 Herlev, Copenhagen, Denmark; hjskovgaard@baan.com.

Two successful applications of salesPLUS

salesPLUS has been successfully applied to many applications, such as the discrete manufacturing of pumps, robots, and vehicles; telecommunications; and high-tech services. We'll now focus on two real-world applications.

Audiovisual-system configuration

Bang & Olufsen manufactures high-end consumer audiovisual equipment (Figure A shows an example). Their products are marketed globally by approximately 1,500 prequalified retail outlets. Their 1996 turnover was approximately $450 million. In cooperation with a number of universities and other European organizations, B&O developed the initial concepts behind salesPLUS.

The problems. B&O's audiovisual systems suffered from product complexity. This led to a long delivery time and lower dealer loyalty because of fear of market competition. Dealers could not handle the configuration complexity in the short delivery time that customers demanded. Also, order errors were normal; dealers usually sold incompatible components or were missing components. The transportation and resupply costs associated with these errors turned profitable sales into losses.

The objectives. B&O wanted to

- increase and maintain dealer loyalty,
- maintain or reduce dealer-education requirements,
- improve order accuracy and completeness,
- lower logistical costs and obtain the correct data at all levels, and
- produce systems that are easy to learn and use.

The solution. B&O implemented salesPLUS and distributed it to their dealers around the world, in seven different languages. The entire Beolink range of products and accessories, as well as equivalent parts to systems up to 10 years old, were modeled in salesPLUS. Updates of the product model are now distributed quarterly, and salesPLUS transmits the orders directly to order entry. salesPLUS tailors and produces customer and service documentation for sold systems. Ongoing model maintenance is approximately one-half of a man-year per year for all product ranges.

Figure A. A Bang & Olufsen audiovisual system.

The results. B&O's implementation of salesPLUS has directly and indirectly produced these improvements:

- Average speaker sales have increased from 2.1 to 2.8 per system.
- Errors in orders have decreased by 4%, saving $5.6 million per year.
- Electronic distribution of sales and product documentation through salePLUS has saved more than $300,000 per year.
- The use of service personnel has decreased, saving more than $130,000.
- Stock levels have decreased from $50 million to $12.5 million.
- Education days have decreased from 5,000 to approximately 2,500 per year.
- Total sales have increased, reflecting higher dealer loyalty.

Future plans. The system has been enhanced into a multimedia configuration and sales tool and will soon be launched as an interactive selling tool. That is, the customer will be able to directly access the system to configure his or her products. Also, B&O plans direct electronic transmission of information such as bills of materials from sales-order

Figure B. A Wittenborg drink-vending machine.

entry to production-engineering and MRP (manufacturing resource planning) systems. There will be no manual transmission from sales business to manufacturing. In addition, we will help B&O to integrate salesPLUS with existing computer systems, working toward an integrated product-development environment.

Drink-vending-machine configuration

Wittenborg A/S manufactures and distributes drink and small-goods vending machines (see Figure B). They distribute them to the Danish and Norwegian markets directly and, through subsidiaries, to the UK, German, French, Dutch, and other Nordic markets. Their 1996 turnover was approximately $125 million.

Wittenborg faces the challenge that more and more cheaper products are coming to the drink-vending-machine market. Rather than attempting to match on price, they decided to create a strategic advantage through the introduction of new models. These models would be highly flexible and adaptable to the customer's needs, in line with the concept of mass customization. This created a number of new challenges.

The problems. Wittenborg's new vending machines were more complicated to produce than previous models, and no limitations were placed on the sales staff as to the available combinations of options. Five checking steps existed, including the order-entry step, to verify the configuration information for the separate manufacturing stages. However, no overall check ensured that the information was valid for the entire configuration. So, orders that could not be manufactured often passed through to production.

The maintenance of parts lists for the vending machines became a huge burden for the organization, because of the mass-customization approach. Each new variation of vending machine required the creation of a new parts list for use during production. This, combined with the ongoing maintenance of the large number of parts lists, was becoming unmanageable.

The front-panel signs for drink selection, the internal hardware setup and placement, and the supporting software parameters had to be configured. The signs were not the same in all languages, which constrained the possible valid combinations, depending on the destination country. The number of ingredients as inputs to the drinks was limited, which added another complicating constraint. The internal drink-production hardware setup had to be drawn using CAD software, to enable production to correctly manufacture the unit. This process alone took approximately 15 hours' work over a one-week period. In addition, the software parameterization for options such as drink recipes, temperatures, and display languages had to be matched and coded by hand into the finished vending machine, using a special terminal. This process was time-consuming and error-prone.

The objectives. Wittenborg wanted to

- eliminate the sales of vending machine configurations that were impossible to manufacture,
- obtain all the data necessary to configure the customer's product,
- minimize the effort in developing new and maintaining existing products,
- minimize the effort in producing and maintaining product documentation,
- minimize the effort in converting an order to a manufacturing process,
- provide accurate and timely information to the assembly line, and
- minimize errors in data through electronic interfaces.

The solution. The Order Entry department now configures orders for vending machines in salesPLUS. The system was also used to tightly integrate a newly developed graphical application to handle the layout of the front-panel signs. salesPLUS is used in conjunction with the sign application to generate a features and options list including production routing information, and diagrams (graphics files) for assembly. salesPLUS passes this information into the Navision Finance System for allocation of an order number, for invoicing, and so on. Navision passes the necessary production information and diagrams to the Mapics MRP system to commence the manufacturing process.

A Microsoft Access database stores the salesPLUS configuration data. The configuration data associated with the correct order number is stored in a parameter file used for configuring the vending machine's internal software. A special industrial terminal is used to match the order number with the appropriate parameter file, which is then uploaded directly into the vending machine.

The product modeling required approximately four man-weeks' work, with ongoing plans estimating around 20% of a man-year of work per annum.

Wittenborg has used salesPLUS since May 1996. They have implemented it for a single product that represents 15% of turnover.

The results. The implementation of salesPLUS, combined with the tight integration with back-end systems, has produced these improvements:

- Maintaining the completeness of order information has saved approximately $80,000.
- Order-configuration time has decreased from 15 hours per week to five minutes per day.
- The work required in sales and order-entry processes has decreased, saving approximately $80,000.
- Logistical errors have decreased significantly, saving approximately $160,000.
- Four manual steps (checking, drafting, variant definition, and routing definition) have been eliminated, through the combination of salesPLUS functionality and electronic integration with the finance system and MRP system.
- All the materials required for production are accurate and available together.
- The ongoing maintenance of product, bill of materials, and routing definitions in the MRP system has greatly decreased.
- Over 400 possible variations can be configured and supplied within nine months.
- Shorter total delivery times and product flexibility have increased sales.

Future plans. Wittenborg intends to use salesPLUS in the short to medium term to realize additional benefits. They plan to

- implement the full product range (15 products) into salesPLUS;
- eliminate all paper-based sales-catalog materials by using salesPLUS' multimedia capabilities; and
- train the sales staff and subsidiaries in the use of salesPLUS, thus enabling interactive selling and eliminating order-entry requirements.