Mass customization and configuration: requirement analysis and constraint based modelling propositions

Michel Aldanondo, Khaled Hadj-Hamou, Guillaume Moynard, Jacques Lamothe

▶ **To cite this version:**

Michel Aldanondo, Khaled Hadj-Hamou, Guillaume Moynard, Jacques Lamothe. Mass customization and configuration: requirement analysis and constraint based modelling propositions. Integrated Computer-Aided Engineering, 2003, 10 (2), pp.177-189. 10.3233/ica-2003-10207 . hal-00452955

HAL Id: hal-00452955
https://hal.science/hal-00452955v1

Submitted on 13 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mass customization and configuration: Requirement analysis and constraint based modeling propositions

Michel Aldanondo[a], Khaled Hadj-Hamou[a], Guillaume Moynard[a,b] and Jacques Lamothe[a]
[a]*Centre de Génie Industriel, Ecole des Mines d'Albi-Carmaux, Campus Jarlard, 81013 Albi CT Cedex 09, France*
Tel.: +33 0 5 63 49 32 34; Fax : +33 0 5 63 49 31 83;
E-mail: {Michel.Aldanondo, Khaled.Hadjhamou, Guillaume.Moynard, Jacques.Lamothe}@enstimac.fr
[b]*Lapeyre-GME, 2-4 rue André Karman, 93200 Aubervilliers, France*
E-mail: guillaume.moynard@lapeyre.fr

Abstract. The purpose of this paper is, on the one hand, to identify to define and classify customization requirements and, on the other hand, to evaluate how generic modeling and configuration assistance within the Constraint Satisfaction Problem (CSP) framework can fulfil the requirements. The aim is to provide commercial configurator knowledge base designers with constraint based generic modeling elements for customizable industrial product. A first part recalls the main trends of the configuration problem. In a second part divided in four sections corresponding with different requirement set; each section proposes a definition of the requirement set, some CSP based modeling elements and a discussion about adequacy of relevant configuration assistance techniques.

1. Introduction

In order to improve their competitiveness, companies try to launch on the market products with customization capabilities. Therefore, they include configuration software (configurator) in their information system. In consequence, most of the ERP providers include in their software packages configuration modules (see the surveys in [18]). In order to work, these configuration modules require a generic model of the product, which is most of the time not defined by a computer science specialist. Therefore, a strong need for friendly generic modeling approaches and relevant tools is rising in order to set up configurators in industry.

Many papers dealing with configuration are more interested by a solving approach and the relevant generic modeling problem is most of the time not addressed. As Wielinga explains in [27] "Many authors prefer a parsimonious set of knowledge structures that suit their favorite problem solving method or their domain". Oriented towards configuration requirements and modeling, our contribution targets two goals. The first one, dealing mainly with product generic modeling requirements, is to identify and to classify the product customization possibilities that should be fulfilled by configurators. The second one, dealing with modeling, is to analyze how the Constraint Satisfaction Problem (CSP) framework can handle these modeling requirements and to comment how existing processing approaches can assist the configuration process.

We first recall configuration definition basics, discuss about the people involved in the configuration process, point out two kinds of configurators, underline general modeling needs and justify our CSP based modeling choice. Then we identify a first set of modeling requirements corresponding with what we call the "central problem" and provide modeling elements. Then, in order to match other particularities of industrial problems, we sequentially add modeling requirements to this "central problem" and analyze how CSP based generic modeling elements can fulfill them. These complementary elements gather: product de-

scriptive approach, tailored or parametric component, layout definition and hierarchical bill-of-materials.

Our intention is therefore to propose and discuss generic modeling elements without focusing on problem solving. We are clearly interested in identifying requirements and showing how CSP can be used to represent various aspects of product generic modeling for the configuration problem. A "custom storage system" provided by the Lapeyre Company will be used as an example to illustrate our ideas all through this paper.

2. General aspects of configuration, configurators and modeling

This section recalls general aspects of configuration, introduce the product generic modeling need and justify our CSP based approach.

2.1. Configuration and configurator elements of definition

From significant works achieved concerning configuration [7,12,15,22], and [1], common features defining configuring are:

- hypothesis: a product is a set of components,
- given: (i) a generic model of a configurable product able to represent a family of products with all possible variants and options, in which a generic model is a set of components plus a set of various constraints; and (ii) a set of customer requirements, in which each requirement can be expressed by a constraint,
- configuring can be defined as "finding at least one component set that satisfies all the constraints".

A configurator is a software package that assists the person in charge of the configuration task. It is composed of a knowledge base that stores the generic model of the product and a set of assistance tools that helps the user finding a solution or selecting components. In any case, the fundamental assistance requirement is to guarantee the consistence of the configured product with both generic model and customer requirements.

2.2. Two classes of configuration situations

According to the previous definition, configuration can exist in any customer/supplier relation when defining the product object of a deal. It is nevertheless possible to identify two main classes of configuration problems and relevant configurators.

The first class deals with the selling side and is conducted by the sales teams of companies. In that case, the cycle time order of magnitude for setting a solution is less than an hour and sometimes less than a minute when configuring on line on a web site for example. This is the target of the ERP configuration modules that assist the Business to Customer (B2C) customer/supplier relation and need to be able to support mass configuration or a great amount of configuration tasks (for example: 1 to 100 configuration tasks per day).

The second class is close to a product design activity and is mainly achieved by the research and development teams of companies. An order of magnitude of the cycle time to provide a solution could be between a week and a month. In most of these cases, previous ERP configuration modules can not be used and specific configurators are necessary. For that class of problems, the customer/supplier relation is more on the Business to Business side (B2B) and the number of configuration tasks that needs to be achieved is rather small (no mass configuration, for example: from 1 to 10 configuration tasks per month).

As the configurators of the second class are specific, each of them (including its generic model) needs to be designed and maintained by computer science specialists. On the other hand, the configuration modules provided by software companies should propose (and provide most of the time) a generic modeling environment enabling a non-computer science specialist to describe the generic model of the product. Our contribution is clearly positioned on the generic modeling side of the first configurator class, frequently called commercial configurator or sale configurator.

2.3. Generic product model main characteristics

In this section we list and briefly explain the main characteristics that should fulfil the generic model of the product. They will be used in the next section for justification of our a priori choice of modeling formalism.

2.3.1. Easy modeling

The product model and the modeling task should rely on "natural" or easy to understand concepts. The configurable product model should be clearly separated from the configuration process. A graphical representation of the model is a necessity for an easy com-

prehension and a lower maintenance effort. Pieces of "visible" written code or pseudo-code (whatever the language is), source of many errors during modeling, should be avoided as much as possible.

2.3.2. Generic model expressivity and knowledge level of users

The generic model can be different according to the configurator user's level of knowledge about the product that must be configured. For a product expert, a generic model gathering detailed components plus constraints is suitable, whereas a clearer product representation in terms of product characteristics is necessary to a non expert. In the work of [14] this is addressed as an explicit model (component oriented) versus an implicit model (description oriented).

2.3.3. Generic model testability

As generic models mainly rely on components plus constraints, it is necessary to have a good confidence in the model consistency. This is a hard point and the configuration technique operating on the model should be open to debugging and inconsistency detection techniques.

2.3.4. Structured generic model and reusability

In order to avoid redundancy of parts of generic model, a structured generic model is of interest and permits to re-use these generic model parts in generic models of different products.

2.4. Model characteristics, configuration techniques and CSP approach

Most of the works recently achieved on configuration rely on propositional logic, first order logic and constraint satisfaction problem (CSP) frameworks. Generic modeling with propositional logic requires a large amount of Boolean variables and Boolean rules (and, or, not), resulting models are therefore complicated with a low expressivity. First order logic is a good generic modeling tool but is not easy to handle by non computer science specialists, some work has been done [6] to provide some UML based graphical formalism avoiding coding.

The utilization of "pure" CSP, introduced in [13], presents interesting concepts for configuration, but we will see, in Section 3.1.2, why its dynamic extension, the DCSP, proposed in [16] and discussed in [23], is far much better for configuration problems handling.

The clear separation between the model and the propagation or resolution techniques, the concepts of variables/domains/constraints natural for non computer specialist and good graphical representation possibilities make CSP a good candidate for configuration modeling.

The CSP concepts, variables and constraints fit two possible configurable product descriptions: set of components and/or set of characteristics. Furthermore, the work of [21] exploiting the notion of preference in CSP allows CSP based configurator to take into account the preference of the user during the configuration process.

In terms of model testability, some works [11] and [20] have presented some possibilities about CSP model debugging and inconsistency detection, but the proof of full consistence model still relies in the complete analysis of the solution space. In order to avoid this last drawback, consistency restoration and explanation during configuration can be supported by CSP techniques as explained in [3].

CSP approaches do not deal easily with structural aspects. Very little work has been done in this field, composite CSP has been proposed by [17] and some extensions of Dynamic CSP trying to match this aspect have been discussed in [25]. As far as we know, this is clearly not a good point for CSP utilization in configuration.

The kinds of product where various aspect of CSP based configuration have been studied are diverse. A car configuration problem introduced in [16] and discussed by [19,20], and [25], an industrial mixer [23], a machining operation [8], and an automotive wiring system [2] are typical examples of product where CSP based configuration has been used. For these works, it must be noted that: the car and industrial mixer product model examples have been used to illustrate theoretical points of CSP based processing, while the goal of the works dealing with machining operation and automotive wiring system examples was to have a product specific operating configurator without providing any new theoretical points. An interesting survey of product customization variety has been reported [22] but without CSP based support. Our contribution, therefore aims to study the diversity of product customization problems, or product modeling requirements, and how it can be handled with CSP based elements. This is the aim of the rest of the paper.

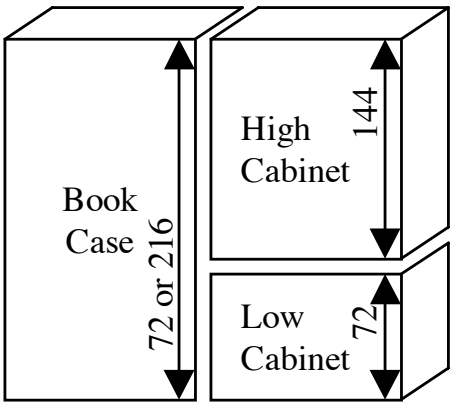


Fig. 1. Basic problem.

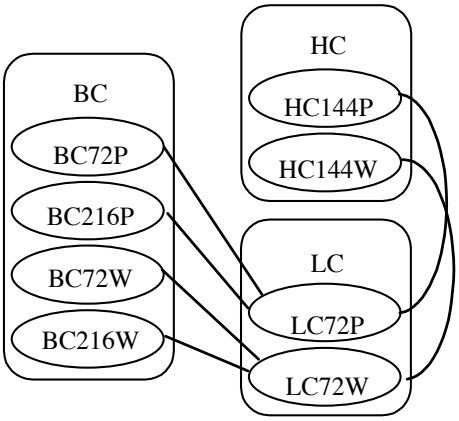


Fig. 2. Basic problem CSP model.

3. Modeling requirements and generic modeling elements

In this chapter, we first present what we call the "central problem", then each following section will describe some additional modeling requirements to the central problem. For each section, the configuration problem hypotheses are defined or refined, an example is given for clarification and a CSP based model is provided and discussed.

3.1. From basic to central configuration problem

3.1.1. Basic configuration problem and CSP approach

The simplest configuration problem can be defined as follows:

- h1: all the components are "standard" or completely defined, it is not possible to create a new component during the configuration task,
- h2: the components are gathered in groups, each component must belong to only one group, the purpose of the group notion is to gather components that support the same functionalities,
- h3: each group is present in any configured product,
- h4: the constraints represent the allowed combinations of components,
- h5: the customer or configurator user requirement corresponds with the selection of one component in each group,
- h6: a configured product is a component set, satisfying both constraints and requirements, where one and only one component must be selected in each group.

The example of Fig. 1, a "custom storage system", illustrates this problem. The generic product gathers 3 components: a Bookcase (BC), a High Cabinet (HC) and a Low Cabinet (LC), where:

- any component of each group exists in two finishes: Painted (P) or Wood (W),
- the BC height can be 72 cm or 216 cm, LC height is 72 and HC height is 144,
- 3 groups exist (i) BC: {BC72P, BC72W, BC216P, BC216W}, (ii) LC: {LC72P, LC72W}and (iii) HC: {HC144P, HC144W}
- a single constraint states that any configured product must be of the same color.

CSP, defined in [13] as a triplet X, D, C where X is a set of variables, D a set of finite domains (one for each variable) and C a set of constraints (defining the possible combinations of variables value), matches this basic problem. Each group of components is associated with a variable. Each component corresponds with one value of the variable. The constraint (solid lines of Fig. 2) represents the allowed combinations of components. A generic model of the product of example 1 could be as the one shown in Fig. 2.

3.1.2. Central configuration problem and DCSP approach

This previous problem is extremely rare. Very frequently, the existence of a group of components is optional or restricted for product feasibility reason or customer wishes. Therefore, two kinds of groups must be defined: the groups which always exist in any configured product and the ones that may exist according to the customer requirements or product feasibility reasons. The central configuration problem can therefore be described as follows:

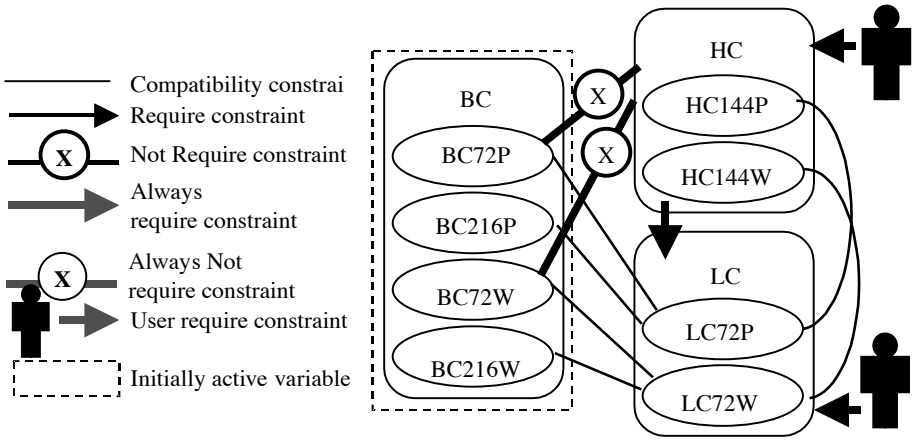


Fig. 3. Central problem DCSP model.

- h1, h2, h6: unchanged
- h3 – a group is either always present in any configured product or its existence depends on: (i) the existence of other groups and/or (ii) the selection of other components,
- h4 – the constraints represent the allowed combinations of (i) components and/or (ii) group existences,
- h5 – the customer or configurator user requirement corresponds to the selection of (i) one component in each group and/or (ii) decision of a component group existence.

  The example of 3.1.1 is modified with the addition of the following constraints:
- The Bookcase must be present in all configured Storage Systems.
- The High Cabinet can exist if and only if a Low Cabinet exists and the Bookcase is 216 cm high.

This "central problem" is supported by most of the configurator packages provided by software companies and is frequently associated with what is called "pick to order" or "assemble to order" industrial situations.

DSCP, proposed in [16], adds to "pure CSP" the notions of:

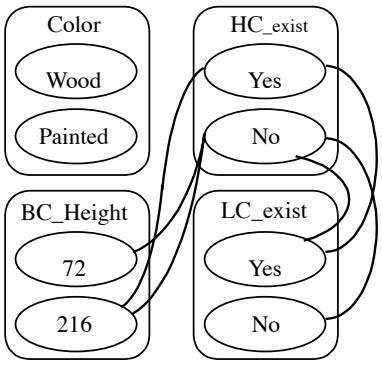
- Initial variables: variables that exist in any configured product.
- Compatibility constraints: equivalent to the constraints defined in Section 3.1.1.
- Activity constraints: allowing to control the variable existence in the following ways:
  - Require: a specified value of a variable "x" implies the existence of the variable "y",
  - Always Require: any value of a variable "x" (or "x" exists) implies the existence variable "y",
  - Not Require: a specified value of a variable "x" implies the non existence of the variable "y",
  - Always Not Require: any value of a variable "x" (or "x" exists) implies the non existence of the variable "y".

A generic model of the central problem example could be as the one shown in Fig. 3. It is clear that DCSP matches exactly the "central problem" requirements. In the following sections we are going to show that the presented requirements are not sufficient to take into account various industrial situation needs and provide complementary requirements and analyze how DCSP can fulfil them.

3.2. Central problem extension and complementary modeling elements

3.2.1. Physical versus descriptive model

The previous model of Fig. 3 is a "physical" model because each variable corresponds with a group of components. But very often, for a non product expert, it is necessary to define "descriptive" attributes that do not correspond with a component group. In that modeling approach, the values of the descriptive attributes permit to identify a component (for example the height and the color allow identifying a component). The advantages are, on the one hand, that descriptive attributes (defined by the person in charge of modeling) can be much more expressive for the customer than a list of components (that does not interest the customer most of the time) and, on the other hand, that configuration models can involve much less configuration variables and constraints. But the second approach needs to maintain identification tables or functional constraints



Color	BC_Heig.	Component
Wood	72	BC72W
Wood	216	BC216W
Painted	72	BC72P
Painted	216	BC216P

Color	HC_exist	Component
Wood	Yes	HC144W
Painted	Yes	HC144P

Color	LC_exist	Component
Wood	Yes	LC72W
Painted	Yes	LC72P

Fig. 4. Central problem descriptive model.

allowing to derive the component list from the values of the descriptive attributes.

The example of Fig. 3 is presented in Fig. 4 with this second modeling approach with the same DCSP formalism. The four descriptive attributes, necessary to represent exactly the same set of solutions, correspond with the color (wood/painted) the book case height (72/216) the existence of the lower cabinet and higher cabinet (yes/no). The component identification tables permit to derive, from the four previous variables, the final result of configuration as a set of BC, LC and HC components.

3.2.1.1. Extended definition of the central problem. In the first definition, components were gathered in groups. With the descriptive approach we added descriptive attributes and values. We therefore introduce the notion of configuration variables that gathers these two elements and can propose the following definition:

- H1 – All the components, "standard" or completely defined, are gathered in groups, each component must belong to only one group. Each group is associated with a configuration variable whose definition domain is a list of values equal to the component list.
- H2 – A product can be characterized by descriptive attributes. Each descriptive attribute is associated with a configuration variable whose definition domain is a list of values, where a value cannot be a component.
- H3 – A configuration variable is either always present in any configured product or its existence depends on: (i) the existence of other configuration variables and/or (ii) the selection of other configuration variable values.
- H4 – The constraints represent the allowed combinations of (i) configuration variable values and/or (ii) configuration variable existences,
- H5 – The customer or configurator user requirement corresponds to the selection of (i) one value for each configuration variable and/or (ii) decision of a configuration variable existence.
- H6 – A configured product is a set of configuration variable values, satisfying both constraints and requirements, where (i) one and only one component is selected in each group and (ii) one and only one value is selected for each existing descriptive attribute.

3.2.1.2. Discussion. Globally the descriptive model targets to displace the configuration from a component plus constraint problem to a descriptive attribute plus constraint problem plus component identification tables. This permits a kind of disconnection between the product configuration process and the bill-of-materials generation. The interest of this disconnection lies in a kind of delinking of selling and manufacturing concerns allowing remote configuration possibilities for sale without handling all the product technical data. Therefore this approach is of interest for commercial configurators.

When there is not any descriptive attribute (i.e. all configuration variables represent component groups), this definition corresponds with the physical problem of Section 3.1.2. When all the configuration variables associated with the component groups are only present in component identification tables or functional constraints, this definition corresponds with a "pure" descriptive problem. Most of the time, it is unfortunately necessary to express the requirements of the configurator user in terms of both component selection and product characteristic valuation. For example, when it is necessary to allow configuration by product experts (component selection) and also by non product experts (descriptive attribute valuation), the two modeling ap-

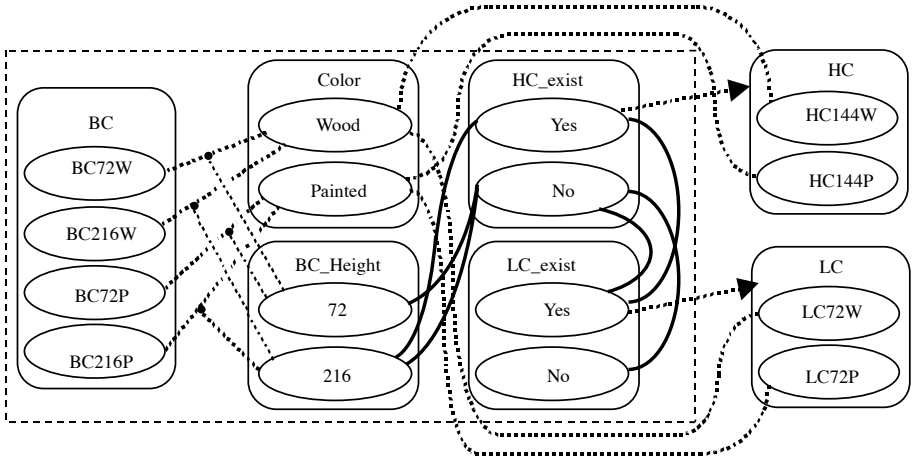


Fig. 5. Physical and descriptive configuration model.

proaches need therefore to be mixed in a single configuration problem as shown in Fig. 5, where both component groups (constraints in dot lines) and descriptive attributes (constraint in solid lines) are present in the proposed model. D CSP handles this problem relying on configuration variables gathering group/component and descriptive attribute/value. The next sections will identify other kinds of attribute that will be associated with configuration variables allowing to represent other configuration modeling requirements.

3.2.2. Tailored components, component quantity and numerical constraints

3.2.2.1. Standard and tailored components. Until now, all the components were completely defined with entirely frozen characteristics (hypothesis H1). Very frequently, industrial cases need to deal with parametric or tailored components. For example, when a door needs to be replaced in an old house, it rarely corresponds with a standard offer. In order to capture this market, many companies propose customization possibilities that are not restricted to standard component assembling.

We therefore propose to characterize each tailored component by numerical tailoring attributes with a continuous definition domain defining the range of possible values. Therefore, modeling requires to associate each tailoring attribute with a configuration variable defined on a continuous definition domain (Fig. 6).

In the example of Fig. 6, the Bookcase and the High Cabinet are now tailored components with tailoring attributes BC_height (chosen in a range between 72 cms and 216 cms) and HC_height (chosen in a range between 50 cms and 144 cms).

3.2.2.2. Component quantity. In the central problem, hypothesis H6 assumes that a configuration solution contains only one component in a group. Very often, it is necessary to model that the quantity of a selected component might be different from 1.

In that case, a quantity attribute is necessary to characterize the quantity selected for a component chosen in a group. This attribute can be defined either on a discrete or continuous domain. This quantity attribute must also be associated with a configuration variable.

In our example of Fig. 6, the High cabinet can contain drawers and roll_out-shelves. For each of them, it is possible to select a quantity between 0 and 3 thanks to the quantity attributes for drawers (Qtt_Draw) and roll-out-shelves (Qtt_Ros).

3.2.2.3. Numerical constraints on discrete variable. Until now, compatibility and activity constraints were discrete. Activity and compatibility constraints represented allowed the combinations of configuration variable values. Very frequently during modeling, it is simpler to define constraints through the expression of a mathematical formula that avoids a huge combinatory even when dealing with discrete variable domains.

Therefore a strong modeling requirement is to handle constraints expressed with formulae. These kinds of formula allow to take into account among other that some component selections or attribute valuations are restricted by some kind of resource (space, power, length...) provided by other components or attribute values. This behavior is similar to the resource/provide/consume elements introduced by [10] and discussed in [19].

In our example of Fig. 6, let us consider that up to four elements among Drawers and Roll-Out-Shelves

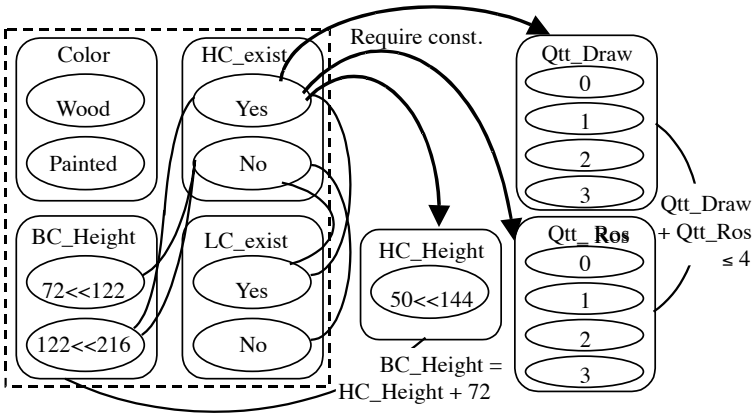


Fig. 6. Tailoring and quantity attributes with constraints.

can be chosen for the High Cabinet. It is much simpler to express this modeling requirement with a numerical constraint stating that Qtt_Draw.+ Qtt_Ros. $\leqslant 4$ than describing in extension the possible combinations of quantity.

3.2.2.4. Numerical constraints on discrete and continuous variable. Before the identification of tailored and quantity attributes, configuration variables were discrete. With these two continuous attributes, it is necessary to express constraints on continuous variables.

A first modeling solution that avoids mathematical formulae is to discretize the variable domains and to use discrete constraints. This is shown in our example of Fig. 6 with the compatibility constraint between BC_height and HC existence, where BC_height is discretized in two intervals [72,122] and [122,216], the first interval forbids the HC existence while the second leaves the two possibilities.

When discretization is not suitable or requires a too large combinatory description, it is necessary to use a mathematical formula. This is shown in the example of Fig. 6 where a constraint, between BC_height and HC_height expresses that the top of Bookcase and High Cabinet must be at the same height: BC_height = HC_height + 72,

3.2.2.5. Tailored attribute, quantity attribute and numerical constraint example. In the example of Fig. 6, the Bookcase and the High Cabinet are tailored components with tailoring attributes BC_height and HC_height, with the following constraints:

- the top of Bookcase and High Cabinet must be at the same height: BC_height = HC_height + 72,
- the High Cabinet can exist if: BC_height $\geqslant 72 + 50 = 122$

Two new component groups, Drawers and Roll-Out Shelves, are added to the High Cabinet. A constraint states that a maximum of four elements among these groups can be chosen. This is modeled with:

- quantity attribute for drawers (Qtt_Draw) and Roll-Out Shelves (Qtt_Ros),
- a numerical constraint expressing that: Qtt_Draw. + Qtt_Ros. $\leqslant 4$.

3.2.2.6. Discussion. As far as modeling is concerned, configuration variables corresponding with tailoring attributes and quantity attributes, numerical constraints, continuous variables and constraints can be added to the model of the central problem. The problem raised in this section is on the side of configuration processing, DCSP solving and constraint propagation require discrete variables and discrete constraints. Therefore, other processing techniques must be investigated for assisting the configuration process, which should deal with cooperation of constraint solvers dealing with discrete and numeric variables, some ideas for this problem have been proposed in [24]. As far as we know, very few commercial configurator software programs support discrete and continuous variables, discrete and continuous constraints, and constraints expressed with formulae in a proper way.

3.2.3. Layout aspects in configuration

In the previous cases, the configuration result is always a set of components plus attribute values. As explained in [4], a same set of components can correspond with two different products according to two dif-

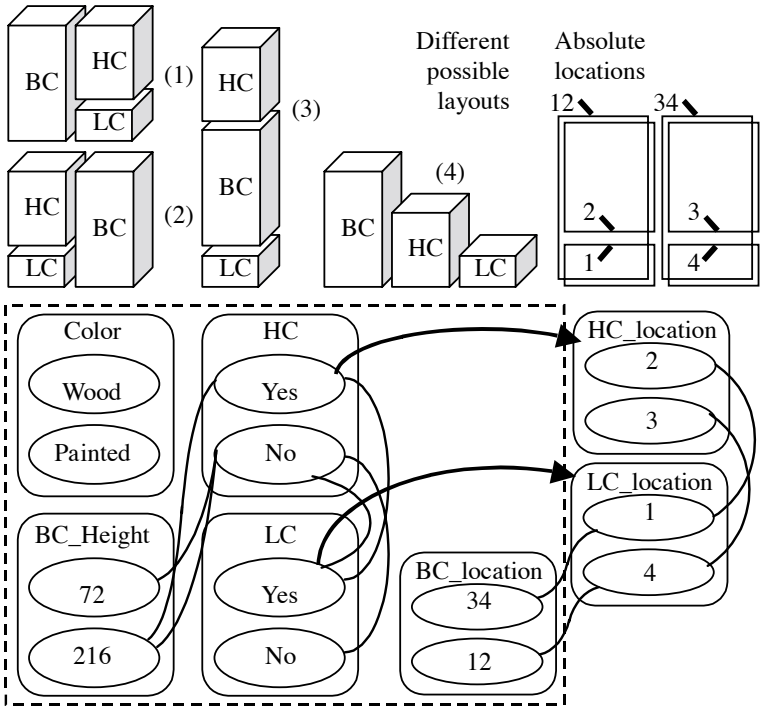


Fig. 7. Layout modeling with location attributes.

ferent layouts. It is therefore sometimes necessary to geometrically locate each selected component among others. This is close to a design activity whose aims are either to locate a component in an absolute referential or to locate the position of one component in relation to another component.

3.2.3.1. Layout with component absolute locations. Layout requirements using absolute position can be handled thanks to the association of each component with location attributes. Each attribute is a coordinate of a referential and corresponds with a configuration variable. The values of the location attribute can be either discrete or continuous. Discrete and continuous layout constraints can link location attributes of different components. These modeling elements are similar to quantity and tailored attributes modeling. Consequently, they lead to the same configuration process drawbacks.

Without layout configuration, the four layouts in the upper left part of Fig. 7 are possible. But let's consider that only the first and second are valid: the LC must be under the HC and the BC near the LC. Then, absolute position layout modeling requires to define different possible component locations. The space is therefore mapped in 6 possible locations identified by a number: 1/lower left, 2/upper left, 3/upper right, 4/lower right, 12/left, 34/right, as described in the upper right of Fig. 7. The model, in the lower part of Fig. 7, shows that each component is characterized by a location attribute and compatibility constraints define the acceptable layout solutions (1) and (2).

3.2.3.2. Layout with component relative locations. Layout requirements using relative location of component couple have been introduced with the notion of "port and connection" in [15], defining that two components can be connected through component ports (port_m of component_i can be connected to port_n of component_j). Each port is a characteristic of a component and can be associated with a component port attribute.

In our example, this approach requires to locate each component with respect to each other at a certain place (on, under, lined up, etc.) in a way very close to CAD systems. Therefore each component must be characterized by some ports, each port corresponds with a particular side (or piece of side) where a port of another component can be "in contact" as shown in the upper part of Fig. 8. Constraints should explain how components could be relatively located.

Layout modeling with relative location or ports is much harder to handle with a CSP based approach

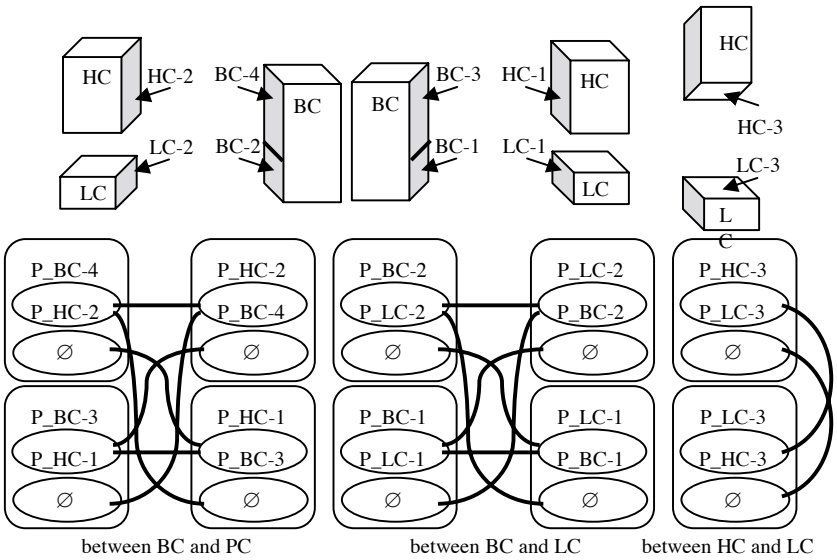


Fig. 8. Layout modeling with ports and connections.

but we propose some ideas dealing with this modeling problem, such as: (i) define a configuration variable for each component port attribute, (ii) define allowed values of the component port configuration variable as the list of component ports that can be connected to it plus the value "Ø" specifying that the port is not used, (iii) define possible connections between component couples with compatibility constraints.

The model in the lower part of Fig. 8 illustrates the proposed solution. The BC has four ports (P_BC-1, P_BC-2, P_BC-3, P_BC-4), the HC three (P_HC-1, P_HC-2, P_HC-3) and LC three (P_LC-1, P_LC-2, P_LC-3). The definition domains of these variables are the component ports that can be connected. The constraints show how three couples of components plus ports can be connected (BC,HC), (BC,LC) and (HC,LC) allowing the solutions of the previous subsection Fig. 7.

3.2.3.3. Discussion. Configuration with layout requirements is not easy to handle with our CSP based elements. When the location problem is not too complicated, for example when locating less than 20 pieces of furniture on a single axis with a unique coordinate, absolute position location is fine. Layout requirements taking advantage of the port and connection approach are typical of electronical product configuration. When components and possible connections are numerous, the presented CSP based model becomes quickly complicated and modeling and maintenance not possible.

It is clear that layout configuration is close to a design activity. Simple cases, close to schematic drawings, can be handled with commercial configurators as shown for example in the configuration problem of a train car layout addressed in [9]. But as explained in [2] configuration including hard layout problems are better solved thanks to the cooperation of configurator and CAD system.

3.2.4. Hierarchical bill-of-materials

Many industrial situations require, mainly for production management reasons, to have the configuration result (the component set) presented as a hierarchical bill-of-materials instead of a single level bill-of-materials.

For the example of Fig. 6 gathering up to five of the components, Bookcase (BC), High Cabinet (HC), Low Cabinet (LC), Drawers (Draw) and Roll-Out Shelves (Ros); the single level bill-of-material is as shown in left part of Fig. 9. The hierarchical bill-of-material presentation need may correspond, for example, with the identification of the following sub-assemblies:

- High Cabinet plus Accessories (HCA) gathering: High Cabinet, Drawers and Roll-Out Shelves,
- High and Low Cabinet (HLC) gathering: Low Cabinet and High Cabinet plus Accessories,

and the resulting hierarchical bill-of-materials would be as shown in the right part Fig. 9.

As this is mainly a presentation requirements, we propose to keep the previous modeling elements (CSP

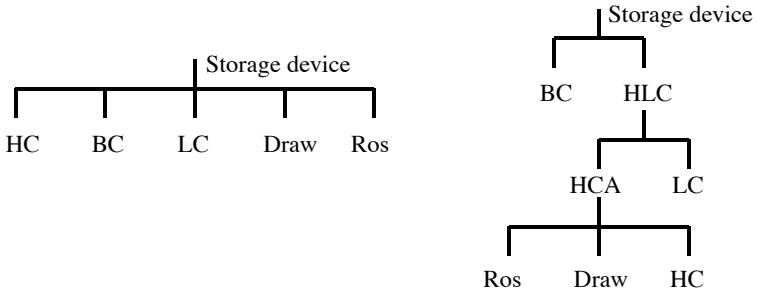


Fig. 9. Single level and hierarchical bill-of-materials.

based generic model plus component identification tables representing single level bill-of-materials) and to add a piece of generic model allowing to derive a hierarchical bill-of-materials. The added pieces of model gather a generic hierarchical bill-of-materials plus existence conditions of each generic sub-assembly. The leaves of the generic hierarchical bill-of-materials are the component groups and the intermediate elements are the generic sub-assemblies of the product. The sub-assembly existence conditions correspond with functional activity constraints that modulate the sub-assembly existence in function of the variables of the CSP based generic model.

With these elements, the configuration can go as follows: (i) configuration of the product (ii) identification of the component (iii) existence validation of the generic sub-assemblies (iv) hierarchical bill-of-material generation. The last step is achieved with a substitution in the generic hierarchical bill-of-materials of each component group by relevant identified component at step (ii). When a sub-assembly does not exist, direct bill-of-material links, between lower level sub-assemblies or components and upper sub-assembly of top level product, replaces the generic hierarchical bill-of-materials upstream and downstream links. Lastly, in order to differentiate each sub-assembly, each of them should be codified according to the lower level components and/or sub-assemblies.

For the example of Fig. 9, these elements both presented in a table form would be as shown in Fig. 10, the hierarchical bill-of-materials in the left part and sub-assembly conditions in the right part

According to these elements, the whole configuration process would go for example as follows:

(i) configuration process with user requirements as: Color = "Wood", BC_Height = 216, LC = "Yes", HC = "yes", HC_Height = 144, Qtt_Draw = 0, Qtt_Ros = 0

(ii) component identification: BC216W (BC group), LC72W (LC group), HC144W (HC group).

(iii) sub-assembly existence validation: HLC,

(iv) hierarchical bill-of-material generation: as shown in Fig. 11.

3.2.4.1. Discussion. The added pieces of the model show how a hierarchical bill-of-materials can be generated as a configuration result with our CSP based approach. Some complementary works are necessary to model a product where a same component group is present in different generic sub-assemblies or when the hierarchical bill-of-materials does not have a tree shape.

A main drawback of our entire CSP based approach, which clearly appears in this section, is that it is not easy to configure in a generative way. Generative configuration should be understood as the ability of configuring a product with more than one instance of a same generic sub-assembly or with a number of instances unknown at the beginning of the configuration task. Very little work has been done in "generative configuration", an application specific configurator has been designed for telecommunication system and is presented in [5] and very recently some ideas about duplicating variable groups in CSP model in order to permit generative configuration have been proposed in [26].

4. Conclusions

Our goal was to identify and classify configuration modeling requirements and analyze how the CSP framework could be a good support for generic modeling and if relevant propagation and solving techniques were adequate.

In terms of identification of requirements and generic modeling, we started with the central problem and a DCSP physical model composed of (i) variables, defined on a discrete domain, associated with component groups, and (ii) compatibility and activity discrete constraints.

Father	Son
Storage device	BC
Storage device	HLC
HLC	LC
HLC	HCA
HCA	HC
HCA	Ros
HCA	Draw

Sub-assembly	Existence condition
HLC	Value(HC) = "Yes"
HCA	Value(Qtt_Draw) + Value(Qtt_Ros) > 0

generic sub-assembly existence condition

generic
hierarchal
bill-of-materials

Fig. 10. Added pieces of generic model for hierarchical bill-of-materials.

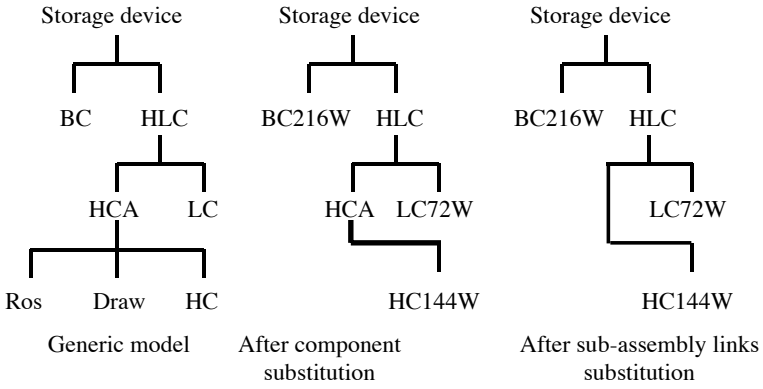


Fig. 11. Hierarchical bill-of-material generation.

We extended the previous central problem, in order to take into account the following generic modeling requirements, and proposed: descriptive approach, parametric or tailored component, multi-presence of a same component, layout configuration and expression of the solution as a hierarchical bill-of-materials. We proposed, for modeling, (i) to associate configuration variables, defined on a discrete or continuous domain, with: component groups, descriptive attributes, tailoring attributes, quantity attributes, location attributes, component ports attributes and generic sub-assemblies; and (ii) constraints, combining the previous variables, that can be compatibility constraints or activity constraints, expressed with allowed combination of values or numerical formulae.

The complexity of the modeling task, with the previous elements, comes from: (i) the mix of all kinds of attributes and the various constraints that can exist in a single problem and (ii) the customization complexity and size of the product itself. Nevertheless, the descriptive approach, allowing some kind of disconnection between product configuration process and bill-of-materials generation, tends to reduce the modeling task difficulty.

The main lack of this approach, at present, lies in the fact that the model is not structured. This forbids easy re-use of model parts without "cut and paste", and makes generative configuration very difficult or impossible when the number of model instances is not known before configuring. This important aspect needs further work with probably some kind of object-oriented approach.

In terms of propagation and solving techniques, continuous domain variables and constraints expressed with mathematical formulae are not compatible with the DCSP solving algorithms and most of the works done in consistency checking, inconsistency explanation proposition, constraint propagation and CSP resolution have been achieved with discrete domain variable and discrete constraints. As far as we know, there is no proper way to overcome the problem at present without a delicate combination of: discretization of continuous variable domains, algorithms reasoning on intervals, or delicate cooperation of different solvers.

Setting an easy to use modeling tool in order to "put on the paper" a generic model of a configuration situation is a big issue for configurator deployment in industry. Friendly user modeling is a necessity, especially for "commercial configurator". The proposed elements present the important interests of being "visible" on a schema and easy to understand by non-specialists thanks to CSP formalism. The model expressivity mixing component groups, various attributes allows to build configuration model that can be used during con-

figuration by product experts (component oriented) and by non product experts (characteristics oriented).

Many of the proposed modeling elements and a specific model structure approach have been included in a commercial configurator software package called "Caméléon Visual Expert" and in a generic modeling method "Caméléon Model Designer" designed and distributed by Access-Commerce (web site: http://www.access-commerce.com). This configurator is also the configuration module of the two ERP Mfgpro and Mapics and has been integrated with many ERP.

Thanks to the elements proposed and the "Caméléon Model Designer" modeling method, a great variety of industrial customizable products that did not require a specific configurator have been successfully modeled. Initially defined for manufacturing products, as the example running all through this paper; the proposed modeling elements can be used for service and software configuration.

References

[1] M. Aldanondo, G. Moynard, K. Hadj Hamou, General configurator requirements and modeling elements, *ECAI Workshop on Configuration*, Berlin, Germany, 2000, pp. 1–6.

[2] M. Aldanondo, J. Lamothe, K. Hadj Hamou, Configuration and CAD modeler: gathering the best of two worlds, *IJCAI Workshop on Configuration*, Seattle, USA, 2001, pp. 1–7.

[3] J. Amilhastre, H. Fargier, P. Marquis, Consistency restorations and explanations in dynamic CSP – Application to configuration, *ECAI Workshop on Modeling and Solving Problems with Constraints*, Berlin, Germany, 2000.

[4] D.C. Brown, Defining Configuring, *AI EDAM* (*Artificial Intelligence for Engineering Design, Analysis and Manufacturing*) **12**(4) (1998), 301–306.

[5] G. Felischandel, G. Friedrich, A. Haselblock, M. Stumptner, Configuring Large Systems Using Generative Constraint Satisfaction, *IEEE Intelligent Systems and their applications* **13** (4) (July/August, 1998), 59–68.

[6] A. Felfering, G. Friedrich, D. Jannach, UML as domain specific language for the construction of knowledge base configuration system, *11th Int Software Engineering and Knowledge Engineering conference*, Kaserlautern, Germany, 1999, pp. 337–345.

[7] G. Freidrich, M. Stumptner, Consistency-Based Configuration, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 35–40.

[8] L. Geneste, M. Ruet, T. Monteiro, Configuration of a machining operation, *ECAI Workshop on Configuration*, Berlin, Germany, 2000, pp. 44–49.

[9] D. Heiderscheilt, H.J. Skovgaard, Visualization of configurations: simplifying configuration user interfaces, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 114–118, Orlando, USA.

[10] M. Heinrich, E. Jungst, The resource-based paradigm: Configuring technical systems from modular components, *IEEE Conf on Artificial Intelligence Applications*, 1991, pp. 257–264.

[11] W. Keirouz, G. Kramer, J. Pabon Principles and Practice of constraint programming Chapter: Exploiting Constraint Dependency Information for Debugging and Explanation, MIT press, 1995, pp. 183–196.

[12] C. Kühn Requirements for Configuring Complex Software-Based Systems, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 11–16.

[13] A. K. Mackworth, Consistency in networks of relations, *Artificial Intelligence* **8** (1977), 99–118.

[14] T. Mannisto, H. Peltonen, R. Sulonen, View to Product Configuration Knowledge Modeling and Evolution, *Technical Report FS-96-03, Workshop on configuration*, AAAI Press, 1996, pp. 111–118.

[15] S. Mittal, F. Frayman, Towards a generic model of configuration tasks, *International Joint Conference on Artificial Intelligence IJCAI*, Detroit, USA, **2** (1989), 1395–1401.

[16] S. Mittal, B. Falkenhainer, Dynamic Constraint Satisfaction Problems, *9th National Conference on Artificial Intelligence AAAI*, Boston, USA, 1990, pp. 25–32.

[17] D. Sabin, E. Freuder, Configuration as Composite Constraint Satisfaction, *Technical Report FS-96-03, Workshop on configuration*, AAAI Press, 1996, pp. 28–36.

[18] D. Sabin, R. Weigel, Product Configuration Frameworks– A Survey, *IEEE Intelligent Systems and their applications* **13** (4) (July/August, 1998), pp. 42–49.

[19] D. Sabin, E. Freuder, Optimization Methods for Constraint Resource Problems, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 83–89.

[20] M. Sabin, E. Freuder, Detecting and Resolving Inconsistency and Redundancy in Conditional Constraint Satisfaction Problems, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 90–94.

[21] T. Schiex, H. Fargier, G. Verfaillie, Valued Constraint Satisfaction Problems: Hard and Easy Problems, *International Joint Conference on Artificial Intelligence IJCAI*, Montreal, Canada, 1995, pp. 631–637.

[22] T. Soininen, J. Tiihonen, T. Männistö, R. Sulonen, Towards a General Ontology of Configuration, *AI EDAM* (*Artificial Intelligence for Engineering Design, Analysis and Manufacturing*) **12**(4) (1998), 357–372.

[23] T. Soininen, E. Gelle, Dynamic Constraint Satisfaction in Configuration, *AAAI Workshop on Configuration*, Orlando, Florida, 1999, pp. 95–100.

[24] C. Tinelli, M. Harandi, Constraint logic programming over unions of constraint theories, *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming* **1118** (1996), 436–450.

[25] M. Véron, M. Aldanondo, Yet another approach to CCSP for configuration problem, *ECAI Workshop on Configuration*, Berlin, Germany, 2000, pp. 59-62.

[26] M. Véron, Modélisation et résolution du problème de configuration industrielle: utilisation des techniques de satisfaction de contraintes, Phd Thesis INP Toulouse, 2001.

[27] B. Wielinga, G. Schreiber, Configuration design problem solving, *IEEE Intelligent Systems and their applications* **12**(2) (March/April, 1997), 49–56.